Data Assimilation in Marine Models

Frydendall, Jan

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

PhD Thesis

Data Assimilation in Marine Models

Jan Frydendall
PhD Student
DTU and DHI
Kgs. Lyngby
IMM-PHD-2009-217

DTU Informatics
Department of Informatics and Mathematical Modeling

Preface

This thesis was submitted at the Technical University of Denmark, Department of Informatics and Mathematical modelling in partial fulfilment of the PhD requirements.
The Ph.D. programme was an ITMAN Graduate school which this study has been a collaboration between the Technical University of Denmark and DHI.
Main supervisors have been Professor Henrik Madsen and Jacob Viborg Tornfeldt Sørensen who have been involved in different parts. During my stay abroad Assistant professor Erik Lindström at Lunds University, Sweden, division of Mathematical Statistics, Centre Mathematical Sciences, also became involved in the study.

Summary

Data Assimilation in Marine Models

This thesis consists of six research papers published or submitted for publication in the period 2006-2009 together with a summary report. The main topics of this thesis are nonlinear data assimilation techniques and estimation in dynamical models. The focus has been on the nonlinear filtering techniques for large scale geophysical numerical models and making them feasible to work with in the data assimilation framework. The filtering techniques investigated are all Monte Carlo simulation based. Some very nice features that can be exploited in the Monte Carlo based data assimilation framework from a computational point of view, e.g. low storage cost, no linearizations of the numerical models, etc. However, this also gives rise to many unforeseen difficulties, e.g. the curse of dimensionality, huge computational costs, etc. The challenge faced in this thesis was finding filters that could handle the nonlinearities encountered in data assimilation and at the same time are robust and reliable enough given the constraints and difficulties that can arise. These problems were addressed in the papers A, E and D.

The other topic of this thesis is estimation in dynamical geophysical numerical models. The challenge of estimating model parameters for well establish geophysical dynamical systems is that these models are not formulated in a way that incorporates the necessary stochastic assumptions that make estimation possible in a maximum likelihood sense. The maximum likelihood approach is selected due to its unique performance in data rich situations.

The estimations are often based on output from the model and the raw observations which lead to suboptimal estimates. The challenge is to give a meaningful description of the model errors through diffusion processes that can be identified and incorporated into the existing maximum likelihood framework. These issues are discussed in paper B.
The third part of the thesis falls a bit out of the above context is work published in papers C, F. In the first paper, a simple data assimilation scheme was investigated to examine the potential benefits of incorporating a data assimilation concept into an atmospheric chemical transport model. This paper deals with the results and conclusions obtained through some of the first experiments with the Optimal Interpolation filter in a geophysical model. The second paper F, deals with the construction of a finite element solver for the Fokker-Planck equation on a 2 dimensional flexible mesh system. The report details the construction of the finite element solver and investigates the potential benefits of a parallel FORTRAN implementation through a series of experiments.

Resumé

Dataassimilering i marine modeller

Denne afhandling består af en sammenfattende rapport samt seks forskningsartikler offentliggjort eller indsendt, med henblik på offentliggørelse i perioden 2006-2009. Afhandlingens vigtigste emner er ikke-lineær dataassimileringsteknikker og estimation i dynamiske modeller. Fokus har været på de ikke-lineær filtreringsteknikker i storskala geofysiske numeriske modeller og anvendelse af modellerne i dataassimileringsrammerne. Alle filtreringsteknikkerne der har været undersøgt, er baseret på Monte Carlo-simulering. Der er mange gode egenskaber, der kan udnyttes i Monte Carlo dataassimilering. Ud fra et computerberegnigs synspunkt kan blandt andet nævnes følgende fordele: lave lagringsomkostninger og ingen linearisering af de numeriske modeller. Dette giver også anledning til mange uforudsete vanskeligheder så som: "Curse of dimensionality", enorme simulerings omkostninger, etc. Udfordringerne i denne afhandling var at finde filtre, som kunne håndtere ikke-lineariteter i dataassimileringsaplikationer og samtidig skulle dataassimileringsværktøjerne også være robuste og pålidelige. Disse problemer er blevet behandlet i artiklerne A, E og D
Det andet emne i denne afhandling er estimation i dynamiske geofysiske numeriske modeller. Udfordringen bestod i at skabe et stokastisk framework til estimation af model parameter i allerede etablere geofysiske dynamiske systemer. Disse modeller er ikke født til at inkludere de nødvendige stokastiske komponenter, som gør det muligt at estimere i en maximum likelihood ramme. Ofte er estimationerne baseret på output fra modellerne og

rå målinger, som fører til suboptimal estimeringer. Udfordringen er at give en meningsfyldt beskrivelse af modelfejlen, ved at beskrive modellerne gennem diffussions processer, som kan identificeres og indarbejdes i eksisterende maximum likelihood rammer. Dette er blevet behandlet i artiklen B
To andre emner som bliver behandlet i afhandlingen, knytter sig mere perifert til de tidligere omtalte studiere. Det drejer sig om arbejdet der er udført i artiklerne C og F. I den første artikel C er et simpelt dataassimileringsskema blevet undersøgt. Undersøglelsen bestod i at efterforske de potentielle fordele ved at indarbejde et dataassimileringskoncept i en atmosfærens kemiske transportmodel. Denne artikel beskæftiger sig med resultater og konklusioner opnået gennem nogle af de første forsøg med et Optimum Interpolations filter i en geofysisk model. Den anden artikel F omhandler implementeringen af en finite element løser for Fokker-Planck ligningen i et 2 dimensionelt system med et fleksiblet mesh. Rapporten beskriver i detaljer konstruktionen af den finite element løser, og undersøger de potentielle fordele ved en parallel FORTRAN implementering samt resultaterne af en række eksperimenter med overnævnte parallel FORTRAN løser.

Contents

Introduction

The main topic of this thesis is nonlinear data assimilation in marine models. The thesis work is a continuation of previous research conducted at DHI and IMM. The background for nonlinear data assimilation is rooted in the geophysical numerical models. The nonlinearities arise from the fact that the real world is highly nonlinear and the governing equations are nonlinear as well. In recent years, data assimilation has proven to be a reliable mathematical method of combing the numerical models with real physical observations. Many of the developed data assimilation techniques have only been developed to deal with linear models and with nonlinear models that could be meaningfully described by means of local linearization schemes. This was not a mathematical barrier, because the methods of handling full nonlinear propagation and updating of the models had been known since the early 1970ties. It was first in the current decade (2000) that computational resources have become available to tackle these obstacles. New computational architectures will change the way numerical models are constructed and simulated. The numerical models have to be prepared for massive multi-core computer that are emerging. The multi-core computers is now available to everyone at low cost. This fact allows for the (further) challenging of nonlinearity problems; this time from an entirely new perspective. This is due to the fact that the problem can now be bombarded with massive Monte Carlo simulations which gives new hope towards understanding the nonlinearities present in data assimilation problems.

Part I

Diffusion Processes

CHAPTER 1

Diffusion processes

Diffusion processes are a class of differential equations which allow for stochastic elements in the formulation as well as in the solution. In this thesis diffusion processes are the basis for all the work presented in the articles and reports. In order to understand the diffusion processes, this chapter gives a short introduction to the concept of stochastic differential equations.

1.1 Ordinary differential equation

In most applications in geophysical dynamical systems, ordinary and partial differential equations are used in almost all model applications. A quick look into any book on geophysics reveals that essentially all system theory is derived by means of differential equations. The complete information about a system at any time point is provided by the so-called state variable X_t. Given the state variable $X_t \in \mathcal{R}^n$ then the derivative of the state variable is given as the function $f(X, t; \theta)$, or more precisely:

$$\frac{dX}{dt} = f(X, t; \theta), \tag{1.1}$$

where $f(X,t)$ is a function of time t and place X, and $\theta \in \mathcal{R}^m$ is the parameter set of the model. With properly given initial conditions this differential equation has a solution.
However, in order to facilitate a proper estimation of the parameters θ based on observations related to the state, X_t a more rigorous formulation is needed.

1.2 Stochastic Differential Equations

It could be tempting just to add a noise term to the above equation, i.e.:

$$\frac{dX}{dt} = f(t, X_t) + g(t, X_t)W_t, \tag{1.2}$$

where $g(t, X_t)$ is a given function and W_t a stochastic process. (Without loss of generality θ has been dropped from the equations for the time being) For the above equation to be meaningful, the stochastic process has to be a Wiener process. From [30] the Wiener process is characterized by:

(i) $P[W(0) = 0] = 1$

(ii) The increments $W(t_1) - W(t_0), W(t_2) - W(t_1), \ldots, W(t_n) - W(t_{n-1})$ are mutually independent for arbitrary time points $0 \leq t_0 < t_1 < \cdots < t_n$.

(iii) For arbitrary t and $h > 0$ the increment $W(t+h) - W(t)$ is normally distributed, with

$$\begin{aligned} E[W(t+h) - W(t)] &= 0 \\ V[W(t+h) - W(t)] &= \sigma^2 h, \end{aligned} \tag{1.3}$$

where σ^2 is the basic incremental variance.

If (1.2) is rewritten as a discrete difference equation [31],

$$X_k = X_0 + \sum_{j=0}^{k-1} f(t_j, X_j)\Delta t_j + \sum_{j=0}^{k-1} g(t_j, X_j)\Delta B_j, \tag{1.4}$$

where it is assumed that the process W_t can be replaced by a process with the following properties: let $0 = t_0 < t_1 < \cdots < t_k = t$ and we have substituted W_t with B_t which is the Brownian motion. It turns out that if the three constraints (i), (ii) and (iii) are to be be fulfilled then it has to be a Wiener process at play, owing to the fact that this process has *stationary independent increments with mean zero* and continuous sample paths.
Taking the limit $\Delta t_j \to 0$, (proof ommitted),

$$X_t = X_0 + \int_0^t f(s, X_s)ds + \int_0^t g(s, X_s)dB_s, \tag{1.5}$$

where we have assumed that the integral $\int_0^t g(s, X_s)dB_s$ exists.
The existence of the integral will be justified without proof, for proof see [25, 31].

1.2.1 Itô Integral

Consider the sum of Wiener processes,

$$\sum_j \varphi(t_j^*, w)B_{[t_j, t_{j+1})}(t)), \tag{1.6}$$

where t_j^* belonged to the interval $[t_j, t_{j+1})$. This sum can unfortunately not be interpreted using the Riemann-Stieltjes sums due to the fact that the Wiener process has unbounded variance. The sum approximation depends on the selection of the point t_j^*. If the point is chosen as $t_j^* = t_j$, the approximation sum is in the limit written as the so-called Itô Integral

$$\sum_j \varphi(t_j, w)B_{[t_j, t_{j+1})}(t)) \to \int_0^t g(s, X_s)dB_s. \tag{1.7}$$

If the starting point was set as $t_j^* = (t_{j+1} - t_j)/2$, the limiting sum:

$$\sum_j \varphi((t_{j+1} - t_j)/2, w)B_{[t_j, t_{j+1})}(t)) \to \int_0^t g(s, X_s) \circ dB_s, \tag{1.8}$$

which is called the Stratonovich integral. However, the focus of this thesis is the integral defined in (1.7) i.e. the Itô integral.
The existence and uniqueness theorem shows that the approximation holds if, $\varphi(s, X_s)$ and $g(s, X_s)$ satisfy the Lipschitz and bounded growth conditions [31].
The Itô integral has some nice properties which are listed below,

$$\int_S^T \varphi(t,w)dB_t(w) = \int_S^U \varphi(t,w))dB_t(w) + \int_U^T \varphi(t,w)dB_t(w), \tag{1.9}$$

$$\int_S^T (c\varphi + \phi)(t,w)dB_t(w) = c\int_S^T \varphi(t,w)dB_t(w) + \int_S^T \phi(t,w)dB_t(w), \tag{1.10}$$

(c constant) for almost all w

$$E[\int_S^T \varphi(t,w)dB_t(w)] = 0 \tag{1.11}$$

$$\int_S^T \varphi(t,w)dB_t(w) = \mathcal{F}_t, \text{is } \mathcal{F}_t \text{ measurable} \tag{1.12}$$

The $\mathcal{F}_t$ is called the filtration. If h is $\mathcal{F}_t$ measurable means that h can be found from the values $B_s(w)$ for $s \leq t$ [31]. That is $h_1(w) = B_{t/2}(w)$ is $\mathcal{F}_t$ measurable, however, $h_2(w) = B_{2t}(w)$ is not $\mathcal{F}_t$ measurable. The conditions $\mathcal{F}_t$ measurable is way of justifying that the integral cannot have any expectations involving future values of the process. The expectation of a stochastic process $\mathbf{X}_t$ given the filtration $\mathcal{F}_{t-k}$ is $E[\mathbf{X}_t|\mathcal{F}_{t-k}] = \mathbf{X}_{t-k}$. The above filtration leads to the fact that Itô integrals are martingales, for more information on martingales and the connection to stochastic differential equation see, [25, 31].
In this section the stochastic differential equation and the Itô integral was defined. However, before continuing with the Itô formula it has to stressed that stochastic differential equation only can be interprid through the choice of stochastic integrals. The choice of stochastic integrals can either be the Stratonovich integral or the Itô integral.

1.3 The Itô formula

The fundamental rule of calculus and the chain rule are the backbone of calculus. The definition of the Riemann integral is not very useful for real calculations. Only on the basis of the fundamental rule of calculus along with the application of the chain rule are explicit calculations possible. It turns out that there are equivalents to the chain rule and the fundamental rule of calculus with stochastic integrals. Before the Itô formula is derived there are still some properties of the Wiener process that have to be considered. From [31] the following statements can be derived. The statements are given without any proof, however, the statements could be derived by the definitions in [31] and through the use of telescope sums.
Consider the incremental Wiener process $\Delta W(t)$ in between two fixed time points s and t, with $t > s$ the increment in time will written as Δt. From the definition on the Wiener process it is known that increment $\Delta W(t)$ is Gaussian distributed and hence it is fairly easy to writte the following:

$$\begin{aligned} E[\Delta W(t)] &= 0 \\ E[(\Delta W(t))^2] &= \Delta t \\ V[(\Delta W(t))] &= \Delta t \\ V[(\Delta W(t))^2] &= 2(\Delta t)^2 \end{aligned} \tag{1.13}$$

Purely heuristic, the squared squared Wiener increment $(\Delta W(t))^2$ has an expectation value of Δt. However, the most important property is the variance of the squared Wiener increment which has the expected value of $V[(\Delta W(t))^2] = 2(\Delta t)^2$. The latter states that if Δt tends towards zero then the expectation of $(\Delta W(t))^2$ will also go towards zero with the variance moving even faster towards zero. This implies that the stochastic differential equation will look more and more deterministic [7].
generalising the above the following important relationships can be found,

$$\begin{aligned} (dB_t)^2 &= dt \\ dt \cdot dB_t &= 0, \end{aligned} \tag{1.14}$$

the first condition come from the generalisation for the Wiener processes in the limit [18]. The second and third condition can be neglected since $(dt)^2$ and the product $dt \cdot dB_t$ are much smaller then dt.
The Itô formula can be derived with a second order Taylor expansion of the Itô process,

$$dX_t = fdt + gdB_t, \tag{1.15}$$

where f and g are adapted processes and let ϕ be a $C^{1,2}$ function. Then define a new Itô process,

$$Y_t = \phi(t, X_t). \tag{1.16}$$

Making a Taylor expansion of (1.16) including second order terms,

$$dY_t = \frac{\partial \phi}{\partial t}dt + \frac{\partial \phi}{\partial x}dX_t + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}(dX_t)^2 + \frac{1}{2}\frac{\partial^2 \phi}{\partial t^2}(dt)^2 + \frac{\partial^2 \phi}{\partial x \partial t}dtdX_t \tag{1.17}$$

With the definitions (1.14) the Itô formula can now be formulated as,

$$dY_t = \frac{\partial \phi}{\partial t}dt + \frac{\partial \phi}{\partial x}dX_t + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}(dX_t)^2 \tag{1.18}$$

This can be considered the chain rule of stochastic Itô calculus. This Itô formula is much easier to work with than the Itô integrals. The formula states that transformation or nonlinear combination of stochastic differentials must be transformed with Itô formula. The normal rules of calculus do not apply for the Itô formulation of the stochastic differentials. The diffusion properties are conserved with the transformation by the Itô formula [26].
Note that for the Stratonovich interpretation (1.8), the normal rules of calculus hold. This is one of the strengths of Stratonovich stochastic calculus. These properties are very nice when working on manifolds with stochastic differential equations and when symmetry properties of the stochastical processes that can be exploited [25]

1.4 The multi-dimensional Itô integral and Itô formula

Multivariate formulation of the Itô integral and Itô formula can easily be generalized from the above theory. In this section the multivariate formulas will be stated. The multivariate formulations are of most importance in this thesis and most of the techniques used in this thesis are derived from the multivariate formulation.

1.4.1 The multi-dimensional Itô integral

Let $B = (B_1, B_2, \ldots, B_n)$ be an n-dimensional Brownian motion and let $g(t, w)$ be a matrices $m \times n$ then:

$$\int_S^T g(t,w)dB_t = \sum_{j=1}^{n} \int_S^T g_{i,j}(t,w)dB_{t,j}(t,w) \tag{1.19}$$

1.4.2 Multi dimensional Itô formula

Let $B = (B_1, B_2, \ldots, B_n)$ be an n-dimensional Brownian motion and let $f_i(t, w)$ and $g_{i,j}(t, w)$ be multi-dimensional processes that satisfy the Lipschitz and the bounded conditions. Then the multivariate Itô process can be defined: [31]

$$\begin{aligned} dX_1 &= f_1 dt + g_{11}dB_1 + \cdots + g_{1n}dB_n \\ &\vdots \quad \vdots \qquad\qquad \vdots \\ dX_m &= f_m dt + g_{m1}dB_1 + \cdots + g_{mn}dB_n \end{aligned} \tag{1.20}$$

If it is further assumed that $\varphi(t, x) = (\varphi_1(t, x), \ldots, \varphi_p(t, x))$ is a smooth map $[0, \infty] \times \mathcal{R}^p \to \mathcal{R}^n$, then the processes: [31]

$$Y(t, w) = \varphi(t, X(t)),$$

is a Itô process whose components are given as,

$$dY_k = \frac{\partial \varphi_k}{\partial t}(t, X)dt + \sum_i \frac{\partial \varphi_k}{\partial x_i} dX_i + \frac{1}{2} \sum_{ij} \frac{\partial^2 \varphi_k}{\partial x_i \partial x_j}(t, X) dX_i dX_j, \quad (1.21)$$

where the following properties are used $dB_i dB_j = \delta_{ij} dt$, $dB_j dt = dt dB_i = 0$.

1.5 Discussion

Diffusion processes play a vital role in data assimilation. Without diffusion processes most of the techniques used in data assimilation would not be possible. Many tend to forget that diffusion processes serve as the corner stones in state filtring as well in estimation in dynamical systems. The Kalman Filter can be applied to dynamical systems without the slightest understanding of the assumed diffusion processes underlying the theory. With the introduction in this thesis, it is hoped that some light will be shed on this very important aspect of data assimilation. In the next chapters, it will be shown how the basic equation for filtring is derived. In the later chapters, the ordinary Kalman Filter will be derived for a dynamical system with partially observed processes. In the same chapter, a brief outlook on other deterministic filters will be given. Finally, the chapter will be concluded with a derivation of the two most important Monte Carlo filters; the ensemble Kalman Filter and the particle filters. Monte Carlo filters are often denoted under the common name of SMC filters, i.e. sequential Monte Carlo filters. In the final chapter, Bayesian inference will be approached from a dynamical point of view. Finally the maximum likelihood estimator will be derived on the basis of diffusion processes and from a SMC perspective.

Part II

Fokker-Planck Equation

CHAPTER 2

The Fokker-Planck equation

The Fokker-Planck equation or the Kolmogorov forward equation is the stochastic version of the Liouville's equation: [18]

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial t} + \sum_{i=1}^{d} (\frac{\partial \rho}{\partial q^i} \dot{q}^i + \frac{\partial \rho}{\partial p^i} \dot{p}^i) = 0, \tag{2.1}$$

where ρ is the phase space distribution function, and q and p are the canonical coordinates and conjugate momentum respectively [2]. The Liouville's equation states that we are able to describe the time evolution of the phase space distribution function. Generally speaking the Liouville's equation states that if we have information on the event at time t_0 then we can integrate the Liouville's equation up to time t to get the information on the evolution of the event along the continuous sample path in the phase space. In the heart of stochastic differential equations is the Fokker-Planck equation. The Fokker-Planck equation is a partial differential equation that describes the evolution of transition probability densities of the Markov processes generated by the stochastic differential equation.

2.1 The Markov process

From [22], a continuous parameter stochastic process $\{x_t, t \in T\}$ is called a Markov process if, for any finite parameter set $t_1 < t_2$, and for all real λ,

$$\Pr(x_{t_2}(w) \leq \lambda | x_\tau, \tau \leq t_1) = \Pr(x_{t_2}(w) \leq \lambda | x_{t_1}). \tag{2.2}$$

Writting the above Markov property in terms of density functions:

$$p(x_{t_2}|x_\tau, \tau \leq t_1) = p(x_{t_2}|x_{t_1}) \tag{2.3}$$

Consider the joint density function $p(x_{t_n}, \ldots, x_{t_1})$. Using the definition of the conditional density function and the Markov property (2.3) yields,

$$\begin{aligned} & p(x_{t_n}, \ldots, x_{t_1}) \\ & = p(x_{t_n}|x_{t_{n-1}}) p(x_{t_n}, x_{t_{n-1}}|x_{t_{n-2}}, \ldots, x_{t_1}) \\ & = p(x_{t_n}|x_{t_{n-1}}) p(x_{t_{n-1}}|x_{t_{n-2}}) p(x_{t_{n-2}}, \ldots, x_{t_1}) \\ & = p(x_{t_n}|x_{t_{n-1}}) \ldots p(x_{t_2}|x_{t_1}) p(x_{t_1}) \end{aligned} \tag{2.4}$$

The joint distribution can thus be fully described solely with prior knowledge of the event $p(x_{t_1})$ and the transition probability $p(x_t|x_\tau)$ for $\forall \quad t > \tau \in T$.

Stochastic differential equations are Markov processes. From the general Itô representation of the SDE:

$$X_t = X_0 + \int_0^t f(X_s, s) ds + \int_0^t g(X_s, s) dB_s, \tag{2.5}$$

where it is assumed that f and g are smooth functions. Since X_t is a nonanticiapating (Martingale) function of t, B_t for $t > 0$ is independent of X_t for $t < 0$. The time development of X_t for $t > 0$ is independent of X_t for $t < 0$ given that we know X_0. Based on the aforementioned example, it be can concluded that Itô representation of the SDE is a Markov process.

For a proof of the Markov properties of the Itô representation of the SDE see [31]
In the next sections the Fokker-Planck equation and the Kolmogorov backward equation will be derived for the one dimensional case. In the last the section the Fokker-Planck equation and the Kolmogorov backward equation will be generalised to the multi-dimensional cases.

2.2 The forward equation

Let the $\{x_t\}$ process by descibed by the density function $p(x_t, t)$ and the transition probability density function $p(x_t, t|x_0, t_0)$. Stating with the density function; given a smooth function $\varphi : \mathbb{R} \to \mathbb{R}$ the expectation is defined as:

$$E[\varphi(X_t)] = \int_{-\infty}^{\infty} \varphi(x)p(x_t, t)dx \tag{2.6}$$

and the derivative with respect to time is defined as:

$$\frac{d}{dt}E[\varphi(X_t)] = \int_{-\infty}^{\infty} \varphi(x)\frac{\partial p}{\partial t}(x_t, t)dx \tag{2.7}$$

However, by Itô formula,

$$\begin{aligned}\varphi(X_t) - \varphi(X_0) = \int_0^t [f(X_s, s)\frac{\partial \varphi}{\partial x}(X_s) + \frac{1}{2}g^2(X_s, s)\frac{\partial^2 \varphi}{\partial x^2}(X_s)]ds \\ + \int_0^t f(X_s, s)\frac{\partial \varphi}{\partial x}(X_s)dB_s\end{aligned} \tag{2.8}$$

Assuming the expectation of the above yields:

$$E[\varphi(X_t)] - E[\varphi(X_0)] = E[\int_0^t [f(X_s, s)\frac{\partial \varphi}{\partial x}(X_s) + \frac{1}{2}g^2(X_s, s)\frac{\partial^2 \varphi}{\partial x^2}(X_s)]ds], \tag{2.9}$$

where the latter part of eq. (2.8) vanishes due to theorem (3.2.2) in [31]. Now differentiating eq. (2.9) with respect to time t and remember that the

expectation of a constant is a constant and from the Fundamental theorem of calculus,

$$\frac{d}{dt}E[\varphi(X_t)] = \int_{-\infty}^{\infty}[f(x,t)\frac{\partial\varphi}{\partial x}(x) + \frac{1}{2}g^2(x,t)\frac{\partial^2\varphi}{\partial x^2}(x)]p(x_t,t)dx]. \quad (2.10)$$

If the latter is combined with the differential version of the expectation operator eq. (2.7) then,

$$\int_{-\infty}^{\infty}[(f(x,t)\frac{\partial\varphi}{\partial x}(x) + \frac{1}{2}g^2(x,t)\frac{\partial^2\varphi}{\partial x^2}(x))p(x_t,t) - \varphi(x)\frac{\partial p}{\partial t}(x_t,t)]dx = 0 \quad (2.11)$$

If it is further assumed that the function and probability density function goes towards zero at the boundary, i.e. $x \to \pm\infty : \varphi, p, \partial\varphi/\partial x, p/\partial x \to 0$[1] then the integrant can be interchanged by integration by parts, hence:

$$\int_{-\infty}^{\infty}\varphi(x)[\frac{\partial p}{\partial t} + \frac{\partial(fp)}{\partial x} - \frac{1}{2}\frac{\partial^2(g^2p)}{\partial x^2}]dx = 0 \quad (2.12)$$

Further assumption had to be made on the drift and the diffusion terms to write the integral on its final form. It was assumed that the terms go to zero on the boundary of the domain due to the arbitrary function $\varphi(x)$ [18]. This is a reasonable assumption because inside the domain, the drift and diffusion terms cannot become discontinuous since they were assumed to be smooth functions. It is only possible for them to become discontinuous across the boundary of the domain. Hence, any transitions from the outside to the inside of the domain have been disallowed. Finally the Fokker-Planck equation can be written as:

$$\frac{\partial p}{\partial t} + \frac{\partial(fp)}{\partial x} - \frac{1}{2}\frac{\partial^2(g^2p)}{\partial x^2} = 0 \quad (2.13)$$

The Fokker-Planck equation is a diffusion process where $f(x,t)$ is the drift term and the $g(x,t)$ is the diffusion term with continuous sample paths. The Fokker-Planck equation completely describes the time evolution of a stochastic process X_t once it is solved.

[1] Compact support

2.3 The characteristic operator

The Fokker-planck equation is an Itô diffusion process since it descibes the evolution of transition probability density of the Markov process generated by the Itô equation [22]. We can rewrite (2.13) by using Theorem 7.5.4 in [31]. The characteristic operator is a generalization of the generator of an Itô diffusion cf. Theorem 7.3.3. in [31] By our notation the theorem states, if X_t is the Itô diffusion $dX_t = f(X_t)dt + g(X_t)dB_t$ and $f \in C^2$ then $\varphi \in \mathcal{D}_A$ and the characteristic operator is,

$$A\varphi(x) = f(x)\frac{\partial\varphi(x)}{\partial x} + \frac{1}{2}g^2(x)\frac{\partial^2\varphi(x)}{\partial x^2} \tag{2.14}$$

For proof of the theorems, look in [31]. With these theorems we are able formulate the forward and backward equations.

2.4 The Backward equation

Using (2.14) a reformulation of (2.11) leads to:

$$\int_{-\infty}^{\infty}[\varphi(x)\frac{\partial p(x,t)}{\partial t} - A\varphi(x)p(x,t)]dx = 0 \tag{2.15}$$

The probability measure has L^p norm and it can be shown that [31] the probability measure has L^2 norm, i.e. it is a Hilbert space, and hence the concept of the adjoint operators can be used to describe the characteristic operator $(\phi, A\psi) = (A^*\phi, \psi)$ where A^* is the adjoint of A. This can also be formulated as:

$$\int \phi(x)A\psi(x)dx = \int A^*\phi(x)\psi(x)dx \tag{2.16}$$

The Forward diffusion operator

It can immediately be see from (2.13) that the adjoint operator A^* is, and hence the generator of the Fokker-Planck diffusion operator is:

$$A^*\psi(x) = -\frac{\partial(f(x)\psi(x))}{\partial x} + \frac{1}{2}\frac{\partial^2(g^2(x)\psi(x))}{\partial x^2}. \tag{2.17}$$

If it also assumed that the transition probability density can be written $p(x, y, t)$ [27], then (2.13) for every fixed $x \in \mathbb{R}$ can be rewritten as:

$$\frac{\partial p(x, y, t)}{\partial t} - A^* \varphi(x) p(x, y, t) = 0. \tag{2.18}$$

The Backward diffusion operator

Kolmogorov Backward equation can also be derived with the concept of a generator operator. Start with the definition of the time evolution of diffusion process and take the expectation,

$$\int_{-\infty}^{\infty} [\frac{\partial(\varphi(y) p(x, y, t))}{\partial t} - A\varphi(y) p(x, y, t)] dy = 0 \tag{2.19}$$

where it is again assumed that the test function is smooth and goes to zero on the boundary, also noting that the generator A is a function on x,

$$\int_{-\infty}^{\infty} \varphi(y) [\frac{\partial p(x, y, t)}{\partial t} - A p(x, y, t)] dy = 0, \tag{2.20}$$

for every fixed $y \in \mathbb{R}$, this leads to the Kolmogorov backward equation for determining the diffusion process backwards in time. By setting $\tilde{p}(x, s) = p(x, t_o - s), 0 < s \leq t_0$ hence:

$$\frac{\partial p(x, s)}{\partial t} = -A p(x, s) \tag{2.21}$$

The Fokker-Planck and Kolmogorov Backward and Liuoville's Equations are all special cases of the Chapman-Kolmogorov equation [18]. For more information on the Chapman-Kolmogorov equation see for example [18, 26].

2.5 Multi-dimensional Forward and Backward equations

Without any further discussion the Fokker-Planck and the Kolmogorov Backward equations will be written in the multi-dimensional form. All the

arguments presented in the previous sections are still valid and the multi-dimensional form is just a matter of generalization. First the Forward (2.14) and Backward (2.17) operators in will be stated in the multi-dimensional form respectively: [25]

$$A^*\psi(y) = -\sum_{i=1}^{d} \frac{\partial(f_i\psi(y))}{\partial y_i} + \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{1}{2} \frac{\partial^2(g_{ij}^2(y)\psi(y))}{\partial y_i \partial y_j} \tag{2.22}$$

$$A\varphi(x) = \sum_{i=1}^{d} f_i(x) \frac{\partial\varphi(x)}{\partial x_i} + \frac{1}{2}\sum_{j=1}^{d}\sum_{i=1}^{d} g_{ij}^2(x) \frac{\partial^2\varphi(x)}{\partial x_i \partial x_j} \tag{2.23}$$

Hence the multi-dimensional version of (2.13) and (2.21) are,

$$\frac{\partial(p(x,y;t))}{\partial t} = A^* p(x,y;t), \qquad \text{for}(t,y) \in (0,\infty) \times (R)^d \tag{2.24}$$

$$\frac{\partial(p(x,y;t))}{\partial t} = Ap(x,y;t), \qquad \text{for}(t,x) \in (0,\infty) \times (R)^d \tag{2.25}$$

2.6 Discussion

The Fokker-Planck equation is at the heart of data assimilation through filtering and estimation of dynamical systems. If the partial or ordinary differential equations are treated as stochastic differential equations as described in Section 1.1 of Chapter 1, then it is possible to apply the filters to the dynamical systems found in typical data assimilation applications. It is necessary to augment the differential equations into stochastic versions of themselves. Without the diffusion term the Fokker-Planck equation and hence the filters cannot be properly derived and would lead to suboptimal results. In Chapter 3, the filtering problem will be solved with the results from Chapters 1 and this chapter. In Chapter 4, the Fokker-Planck equation will be used to derive the maximum likelihood estimator for dynamical systems with partly observed processes.

2.6.1 Large scale simulations with the Fokker-Planck equation

In Appendix F, a large scale numerical solver for the Fokker-Planck equation is constructed using the Finite Element Method (FEM). The main goal was to derive a FEM implementation of the Fokker-Planck equation to be used in a tracking experiment [33]. The key idea is to simulate random walks on a large scale basis on a grid with obstacles. The random walks could simulate people, mammals or fish movement on the grid. To simulate random walks, a transition density had be to calculated in order to draw new samples to the random walks. The transition densities were found by integrating the Fokker-Planck equation with prior knowledge on the starting points of the random walks. However, the existing Matlab FEM solver had serious limitations in computational speed and on the resolution of the grid that was addressed in the tracking experiment. Therefore a FORTRAN implementation of the solution of the Fokker-Planck equation via FEM is suggested as descibed in Appendix F. To get the maximum performance of the solver the OpenMP API was used to make a parallel FORTRAN code to support the SMP architecture of the Sun HPC servers at DTU. The results from the FEM solver and the FORTRAN implementation is documented in the technical report in Appendix F.

Part III

Filtering

CHAPTER 3

Filtering

The cornerstone in data assimilation is filtering. The filtering problem of data assimilation has often been formulated as an inverse problem. That is $y(x) = f(x)$, where y is known, and x is unknown, find x. Or alternatively, x is known and y is unknown, find y; the latter is the forward problem and the former is the inverse problem.
In this section the filtering problem will be formulated by means of probability theory which facilitates the Hidden Markov estimation setting. Through the Fokker-Planck equation derived in the last chapter, the ordinary Kalman Filter will be formulated for a dynamic system. Later, an overview is presented on other approximating Kalman Filters for nonlinear dynamical systems. Finally, state-of-the art nonlinear filters will be derived and presented.
In reality there are three kinds of dynamical system to be considered. All three involve observations and a dynamical model. If the model and the observations are given only at discrete time points then this is called discrete-discrete filtering. However, if the model is a stochastic differential equation and the measurements are still discrete then is called continuous-discrete filtering. The last is, of course, when both the model and the observations are continuous, deemed continuous-continuous filtering.

The dynamical systems relates to data assimilation are almost always continuous models with discrete observations. The observations are, e.g., in-situ measurements which have been logged with a time window of 10 min. or remote sensing measurements with a time frame of 24 hours per sweep, etc.

3.1 Continuous-discrete filtering

Consider the multivariate stochastic differential equation (1.20):

$$d\mathbf{X}_t = \mathbf{f}(\mathbf{X}_t, t)dt + \mathbf{G}(\mathbf{X}_t, t)d\mathbf{B}_t, \qquad t \geq t_0, \tag{3.1}$$

where $\mathbf{f}$ and $\mathbf{X}$ are p valued vectors and $\mathbf{G}$ is a $p \times q$ matrix, $\mathbf{B}_t$ is q valued Brownian motion with $E[d\mathbf{B}_t d\mathbf{B}_t^T] = \mathbf{Q}(t)dt$. The discrete observations are given as:

$$\mathbf{Y}_k = \mathbf{h}(\mathbf{X}_k, t_k) + \boldsymbol{\varepsilon}_k; \qquad k = 1, 2, \ldots; \qquad t_{k+1} > t_k \geq t_o, \tag{3.2}$$

where $\mathbf{h} : \mathcal{R}^p \times \mathcal{R}^r \to \mathcal{R}^r$ is the observation operator that relates the states to the observations. The observations noise $\boldsymbol{\varepsilon}_t \in \mathcal{R}^r$ is a white noise process that is not dependent on the past or current states or the system noise. It is assumed that the PDF of $\boldsymbol{\varepsilon}_t$ is known and that the initial distribution of $p(\mathbf{x}_1|\mathbf{y}_0) = p(\mathbf{x}_0)$ is also know along with the transition and observation operator for all $t \geq t_0$.

3.2 Predicting and updating the conditional density

The conditional density can only be evaluated and updated when observations become available. However, since the model is continuous and the observations are discrete, the transition probability of the model has to be known. Fortunately, the Fokker-Planck equation describes the evolution of transition probabilities (predictions).

Prediction

Between observations $t_k \leq t < t_{k+1}$ the conditional density $p(x(t)|\mathcal{F}_{t_k})$ with known initial distribution $p(x(t_0)|\mathcal{F}_{t_0})$ satisfies the Fokker-Planck equation:

$$\frac{\partial p}{dt} = A^* p, \tag{3.3}$$

where A^* is the generator operator of the Fokker-Planck (2.22):

$$A^*(\cdot) = -\sum_{i=1}^{d} \frac{\partial(\cdot f_i)}{\partial \mathbf{x}_i} + \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{1}{2} \frac{\partial^2(\cdot(\mathbf{G}\mathbf{Q}\mathbf{G}^T)_{ij})}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \tag{3.4}$$

Updating

At observations time point t_k the conditional density can be updated,

$$p(\mathbf{x}, t_k|\mathcal{F}_{t_k}), \tag{3.5}$$

from Bayes' rule the conditional density can be rewritten as,

$$p(\mathbf{x}(t_k)|\mathcal{F}_{t_k}) = \frac{p(\mathbf{y}_k|\mathbf{x}(t_k))p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}})}{p(\mathbf{y}_k|\mathcal{F}_{t_{k-1}})}, \tag{3.6}$$

the first term in the numerator can be reduced since the $\boldsymbol{\varepsilon}_k$ is white noise without memory, i.e. $p(\mathbf{y}_k|\mathbf{x}_k, \mathcal{F}_{t_{k-1}}) = p(\mathbf{y}_k|\mathbf{x}_k)$. The denominator can also be rewritten as$p(\mathbf{y}_k|\mathcal{F}_{t_{k-1}}) = \int p(\mathbf{y}_k|\mathbf{x}(t_k))p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}})dx$. Adapting Bayers' rule with the above yields the general update equation:

$$p(\mathbf{x}(t_k)|\mathcal{F}_{t_k}) = \frac{p(\mathbf{y}_k|\mathbf{x}(t_k))p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}})}{\int p(\mathbf{y}_k|\mathbf{x}(t_k)p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}})d\mathbf{x}(\mathbf{t_k})}, \tag{3.7}$$

where the conditional density at an observation time point t_k given $\mathbf{x}(t_k)$ is:

$$p(\mathbf{y}_k|\mathbf{x}(t_k)) = p_{\boldsymbol{\varepsilon}_k}(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}(t_k))) \tag{3.8}$$

with white noise $\boldsymbol{\varepsilon}_k \sim N(0, \mathbf{R}_k)$, i.e.

$$p(\mathbf{y}_k|\mathbf{x}(t_k)) = (2\pi)^{-m/2}|\mathbf{R}_k|^{1/2} \exp\left(1/2\left[\mathbf{y}_k - \mathbf{h}(\mathbf{x}(t_k))\right]^T \mathbf{R}_k^{-1}\left[\mathbf{y}_k - \mathbf{h}(\mathbf{x}(t_k))\right]\right) \tag{3.9}$$

3.3 Derivation of the conditional moments

Having established the general filtering problem for conditional transition density of the dynamical system, then it should be possible to solve the filtering problem for any SDE driven model. The prediction can in terms of the conditional density be generated with the Fokker-Planck equation (3.3) for the time when observations become available, and then, given a new observation, the conditional density will be updated by means of Bayes' rule (3.6).
The major drawback of this method is that the solution to the Fokker-Planck equation is only known in very few special cases. However, it is possible to derive conditional moments from the Fokker-Planck equation. In the linear case where the conditional densities are Gaussian, this would lead to the Kalman Filter if the first and second order moments were calculated, see, e.g. [22]. Regarding the general filtering problem, something similar can be done. However, since the state equation model is nonlinear, an approximation has to be obtain in order to close the equations. This problem is still one of the challenges in deriving filters for advanced systems.

3.4 Moment generating equation

For now, the first two moments will be derived through the use of the Fokker-Planck equation. Consider the expectation integral (3.10) with smooth function $\boldsymbol{\varphi}$ and with the assumptions in (2.11).
The expectation of $\boldsymbol{\varphi}(\mathbf{X}_t)$ can be written as:

$$E[\boldsymbol{\varphi}(\mathbf{X}_t)] = \widehat{\boldsymbol{\varphi}}(\mathbf{x}_{t_k}) = \int_{-\infty}^{\infty} \boldsymbol{\varphi}(\mathbf{x}) p(\mathbf{x}(t)) d\mathbf{x} \tag{3.10}$$

and the derivative with respect to time is defined as:

$$\frac{d}{dt}\widehat{\boldsymbol{\varphi}}(\mathbf{x}_{t_k}) = \int_{-\infty}^{\infty} \varphi(\mathbf{x}) \frac{\partial p}{\partial t}(\mathbf{x}, t) d\mathbf{x}. \tag{3.11}$$

The time derivative of the probability density can be substituted with Fokker-Planck equation (3.3) and integrated by parts: (the subscripts have been dropped for convenience)

$$\begin{aligned}
\frac{d}{dt}\widehat{\boldsymbol{\varphi}}(\mathbf{x}_{t_k}) &= \int_{-\infty}^{\infty} \boldsymbol{\varphi}(\mathbf{x}) A^* p(\mathbf{x}, t) d\mathbf{x} \\
&= \int_{-\infty}^{\infty} \boldsymbol{\varphi}(\mathbf{x}) \left[-\frac{\partial(\mathbf{f}p)}{\partial \mathbf{x}} + \frac{1}{2} \frac{\partial(\mathbf{G}\mathbf{Q}\mathbf{G}^T p)}{\partial \mathbf{x} \partial \mathbf{x}^T} \right] d\mathbf{x} \\
&= -\left[\boldsymbol{\varphi}\mathbf{f}p\right]_{-\infty}^{\infty} + \int_{-\infty}^{\infty} \frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{x}} \mathbf{f} p d\mathbf{x} \\
&+ \frac{1}{2} \left[\boldsymbol{\varphi} \frac{\partial(\mathbf{G}\mathbf{Q}\mathbf{G}^T p)}{\partial \mathbf{x} \partial \mathbf{x}^T} \right]_{-\infty}^{\infty} + \frac{1}{2} \int_{-\infty}^{\infty} \frac{\partial(\mathbf{G}\mathbf{Q}\mathbf{G}^T p)}{\partial \mathbf{x} \partial \mathbf{x}^T} d\mathbf{x} \\
&= -\left[\boldsymbol{\varphi}\mathbf{f}p\right]_{-\infty}^{\infty} + \int_{-\infty}^{\infty} \frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{x}} \mathbf{f} p d\mathbf{x} + \frac{1}{2} \left[\boldsymbol{\varphi} \frac{\partial(\mathbf{G}\mathbf{Q}\mathbf{G}^T p)}{\partial \mathbf{x} \partial \mathbf{x}^T} \right]_{-\infty}^{\infty} \\
&- \left[\frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{x}} \mathbf{f} \mathbf{G}\mathbf{Q}\mathbf{G}^T \right]_{-\infty}^{\infty} + \int_{\infty}^{\infty} \frac{1}{2} \frac{\partial^2 \boldsymbol{\varphi}}{\partial \mathbf{x} \partial \mathbf{x}^T} \mathbf{G}\mathbf{Q}\mathbf{G}^T p d\mathbf{x}.
\end{aligned} \tag{3.12}$$

Assuming that the function $\boldsymbol{\varphi}$ and probability density have compact support, the square brackets are all going to zero. Using the definition of the expectation operator, then (3.12) can be simplified:

$$\begin{aligned}
\frac{d}{dt} E\left[\boldsymbol{\varphi}(\mathbf{X}_t)\right] = \frac{d\widehat{\boldsymbol{\varphi}}(\mathbf{x}_{t_k})}{dt} &= E_p \left[\mathbf{f}(\mathbf{x}, t) \frac{\partial \boldsymbol{\varphi}(\mathbf{x}, t)}{\partial \mathbf{x}} \right] \\
&+ \frac{1}{2} \mathrm{Tr} E_p \left[\mathbf{G}\mathbf{Q}\mathbf{G}^T(\mathbf{x}, t) \frac{\partial^2 \boldsymbol{\varphi}(\mathbf{x}, t)}{\partial \mathbf{x} \partial \mathbf{x}^T} \right],
\end{aligned} \tag{3.13}$$

where the subscript p denotes that the expectation has to be evaluated with respect to the probability density $p(\mathbf{x}, t)$.

Prediction

From (3.13), the general evolution of $E[\boldsymbol{\varphi}]$ is described. Multiplying (3.13) with dt:

$$d\widehat{\boldsymbol{\varphi}}(\mathbf{x}_{t_k}) = E_p\left[\mathbf{f}(\mathbf{x},t)\frac{\partial \boldsymbol{\varphi}(\mathbf{x},t)}{\partial \mathbf{x}}\right]dt + \frac{1}{2}\mathrm{Tr}E_p\left[\mathbf{G}\mathbf{Q}\mathbf{G}^T(x,t)\frac{\partial^2\boldsymbol{\varphi}(\mathbf{x},t)}{\partial \mathbf{x}\partial \mathbf{x}^T}\right]dt \tag{3.14}$$

Updating

Having established how the propagation of the $\boldsymbol{\varphi}(\mathbf{x})$ behaves between observations, it is now possible to use Bayes' rule when a observation becomes available to find the conditional mean and variance. Multiplying (3.7) with $\boldsymbol{\varphi}(x)$ and integrating over x:

$$\begin{aligned}\widehat{\boldsymbol{\varphi}}(\mathbf{x}_{t_k}) &= \frac{\int \boldsymbol{\varphi}(\mathbf{x})p(\mathbf{y}_k|\mathbf{x}(t_k))p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}})d\mathbf{x}_{t_k}}{\int p(\mathbf{y}_k|\mathbf{x}(t_k))p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}})d\mathbf{x}} \\ &= \frac{E_{(\cdot)}\left[\boldsymbol{\varphi}(\mathbf{x}(t_k)p(\mathbf{y}_{t_k}|\mathbf{x}(t_k))\right]}{E_{(\cdot)}\left[p(\mathbf{y}_{t_k}|\mathbf{x}(t_k))\right]}\end{aligned} \tag{3.15}$$

where $(\cdot) = p(\mathbf{x}_t|\mathcal{F}_{t_{k-1}})$ is a reminder that the expectations have to be evaluated with respect to the $p(\mathbf{x}_t|\mathcal{F}_{t_{k-1}})$. The former equation (3.14) is the moment generating function and the latter equation of (3.15) is a moment updating function. In principle, every moment can be calculated using these equations.

Theorem 3.4.1 (The moment generating function)**.** *Let $\boldsymbol{\varphi}(\mathbf{x}(t_k))$ $\mathcal{C}^2$ be the function of a p dimensional vector $\mathbf{x}$. Then in between observations, the prediction $\widehat{\boldsymbol{\varphi}(\mathbf{x}(t_k)}$ is given by,*

$$d\widehat{\boldsymbol{\varphi}(\mathbf{x}(t_k))} = E^t\left[\boldsymbol{\varphi}(\mathbf{x}(t_k))\mathbf{f}^T\right]dt + \frac{1}{2}\mathrm{Tr}E^t\left[\mathbf{G}\mathbf{Q}\mathbf{G}^T\right]dt, \qquad t_k \leq t < t_{k+1} \tag{3.16}$$

and at observations time point t_k the update is:

$$\widehat{\boldsymbol{\varphi}}(\mathbf{x}_{t_k}) = \frac{E_{(\cdot)}\left[\boldsymbol{\varphi}(\mathbf{x}(t_k)p(\mathbf{y}_{t_k}|\mathbf{x}(t_k))\right]}{E_{(\cdot)}\left[p(\mathbf{y}_{t_k}|\mathbf{x}(t_k))\right]} \tag{3.17}$$

The different moments can be derived by substituting $\widehat{\boldsymbol{\varphi}}(\mathbf{x}) \rightarrow \mathbf{x}_t^n$, where n is the degree of the polynomial and corresonding to the order of the moment. The most interesting moments are the first and second order moments. Hence the first order moment, $\boldsymbol{\varphi} \rightarrow \mathbf{x}_t$:

$$\frac{d\widehat{\mathbf{x}}_t}{dt} = \widehat{\mathbf{f}(\mathbf{x},t)} = E_{(\cdot)}\left[\mathbf{f}(\mathbf{x},t)\right] \tag{3.18}$$

The second order central moment can be found just as easily with $\widehat{\boldsymbol{\varphi}}(\mathbf{x}) \rightarrow \mathbf{x}_t^2$ and subtracting the first order moment squared, i.e. $V[x] = E[x^2] - E[x]E[x]$, the covariance is given as:

$$\begin{aligned}
\frac{d\widehat{P}_t}{dt} &= E_{(\cdot)}\left[\mathbf{x}(t)\mathbf{f}^T(\mathbf{x},t)\right] - E_{(\cdot)}\left[\mathbf{x}(t)\right]E_{(\cdot)}\left[\mathbf{f}(\mathbf{x},t)\right]^T && (3.19)\\
&+ E_{(\cdot)}\left[\mathbf{f}(\mathbf{x},t)\mathbf{x}(t)^T\right] - E_{(\cdot)}\left[\mathbf{f}(\mathbf{x},t)\right]E_{(\cdot)}\left[\mathbf{x}(t)\right]^T && (3.20)\\
&+ \mathrm{Tr}E_p\left[\mathbf{G}\mathbf{Q}\mathbf{G}^T(\mathbf{x},t)\right]. && (3.21)
\end{aligned}$$

When a observation become avaliable at t_k the (3.18) and (3.19) can be updated,

$$\widehat{\mathbf{x}}_t^{t_k} = \frac{E_{(\cdot)}\left[\mathbf{x}(t_k)p(\mathbf{y}_{t_k}|\mathbf{x}(t_k))\right]}{E_{(\cdot)}\left[p(\mathbf{y}_{t_k}|\mathbf{x}(t_k)\right]}, \tag{3.22}$$

$$\widehat{P}_t^{t_k} = \frac{E_{(\cdot)}\left[\mathbf{x}(t_k)\mathbf{x}^T(t_k)p(\mathbf{y}_{t_k}|\mathbf{x}(t_k))\right]}{E_{(\cdot)}\left[p(\mathbf{y}_{t_k}|\mathbf{x}(t_k))\right]} - \widehat{\mathbf{x}}_t^{t_k}\widehat{\mathbf{x}}_t^{t_k\,T}. \tag{3.23}$$

3.5 Stochastic Filters

Equations (3.16) and (3.17) are at the corner stones of the general filtering problem. These equations are all derived in [22], which also inspired the above derivation.
In this section, a brief outlook on the stochastic filters is given. First, however, the Kalman Filter is derived as it is the most fundamental and widely used filter.
The Kalman Filter is a linear filter which assumes that the state space model has to be linear and have Gaussian densities. This means that the initial distribution of (3.1) will have to be Gaussian process $\mathbf{x}(\mathbf{t_0}) \sim N(\widehat{\mathbf{x}}(t_0), \mathbf{P}_{t_0})$. Furthermore, it is assumed that the state equation (3.1) and observation equation (3.2) can be written as:

$$d\mathbf{x}_t = F_t \mathbf{x}_t dt + G_t dB_t, \tag{3.24}$$

$$\mathbf{y}_k = H_{t_k} \mathbf{x}(t_k) + \boldsymbol{\varepsilon}_k, \tag{3.25}$$

where F_t and G_t are $p \times p$ and $p \times q$ continuous matrix functions; B_t is q valued Brownian motion with $E[dB_t dB_t^T] = Q(t)dt$ H_{t_k} is a $p \times r$ bounded matrix function and that $\boldsymbol{\varepsilon}_k \sim N(0, R_k)$ is a white Gaussian noise.
In light of the above assumptions all the densities in (3.6) will be Gaussian. Using the fact that densities are Gaussian it is, then, possible to write the densities in (3.6), as

$$p(\mathbf{x}(t_k)|\mathcal{F}_{t_k}) \sim N(\mathbf{x}(t_k), \mathbf{P}_{t_k}) \tag{3.26}$$

$$p(\mathbf{y}_k|\mathbf{x}(t_k)) \sim N(\mathbf{H}_{t_k}\mathbf{x}(t_k), \mathbf{R}_{t_k}) \tag{3.27}$$

$$p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}}) \sim N(\widehat{\mathbf{x}}(t_k), \mathbf{P}_{t_{k-1}}) \tag{3.28}$$

$$p(\mathbf{y}_k|\mathcal{F}_{t_{k-1}}) \sim N(\mathbf{H}\widehat{\mathbf{x}}(t_{k-1}), \mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}}^T + \mathbf{R}_{t_{k-1}}) \tag{3.29}$$

The latter equation comes from the fact that:

$$\begin{aligned} &E[(\mathbf{y}_{t_k} - E[\mathbf{y}_{t_k}|\mathcal{F}_{t_{k-1}}])(\mathbf{y}_{t_k} - E[\mathbf{y}_{t_k}|\mathcal{F}_{t_{k-1}}])^T|\mathcal{F}_{t_{k-1}}] \\ &= E[\mathbf{y}_{t_k}\mathbf{y}_{t_k}^T|\mathcal{F}_{t_{k-1}}] + E[\mathbf{H}_{t_k}\mathbf{x}_{t_k}(\mathbf{H}_{t_k}\mathbf{x}_{t_k})^T|\mathcal{F}_{t_{k-1}}] \\ &= \mathbf{R}_{t_{k-1}} + \mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}}^T \end{aligned} \tag{3.30}$$

Substituting the above definitions back into (3.6) yields:

$$p(\mathbf{x}(t_k)|\mathcal{F}_{t_k}) = \frac{p(\mathbf{y}_k|\mathbf{x}(t_k),\mathcal{F}_{t_{k-1}})p(\mathbf{x}(t_k)|\mathcal{F}_{t_{k-1}})}{p(\mathbf{y}_k|\mathcal{F}_{t_{k-1}})} \tag{3.31}$$

$$= \frac{|\mathbf{R}_{t_{k-1}} + \mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}}^T|^{1/2}}{(2\pi)^{p/2}|\mathbf{P}_{t_{k-1}}|^{1/2}|\mathbf{R}_{t_{k-1}}|^{1/2}} \exp(-1/2\,\{\cdot\}), \tag{3.32}$$

where,

$$\begin{aligned}\{\cdot\} = &(\mathbf{y}_{t_{k-1}} - \mathbf{H}_{t_{k-1}}\mathbf{x}(t_{k-1}))^T\mathbf{R}_{t_{k-1}}^{-1}(\mathbf{y}_{t_{k-1}} - \mathbf{H}_{t_{k-1}}\mathbf{x}(t_{k-1})) \\ &+ (\mathbf{x}_{t_{k-1}} - \widehat{\mathbf{x}}(t_{k-1}))^T\mathbf{P}_{t_{k-1}}^{-1}(\mathbf{x}(t_{k-1}) - \widehat{\mathbf{x}}(t_{k-1})) \\ &+ (\mathbf{y}_{t_{k-1}} - \mathbf{H}_{t_{k-1}}\widehat{\mathbf{x}}(t_{k-1}))^T(\mathbf{R}_{t_{k-1}} + \mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}})^{-1}(\mathbf{y}_{t_{k-1}} - \mathbf{H}_{t_{k-1}}\widehat{\mathbf{x}}(t_{k-1}))\end{aligned} \tag{3.33}$$

The above equation has to be put on the final form of (3.26). In order to do this, there is some matrix algebra and some rearranging of terms, which is not shown here. Consult [34] for the proper matrix relationships.

$$\mathbf{P}_{t_k} = \mathbf{P}_{t_{k-1}} + \mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}}\left[\mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}} + \mathbf{R}_{t_{k-1}}\right]^{-1}\mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}} \tag{3.34}$$

$$\widehat{\mathbf{x}}(t_k) = \widehat{\mathbf{x}}(t_{k-1}) + \mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}}\left[\mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}} + \mathbf{R}_{t_{k-1}}\right]^{-1}\mathbf{d_k}, \tag{3.35}$$

where $\mathbf{d}_k = \mathbf{y}_{t_k} - \mathbf{H}_{t_k}\widehat{\mathbf{x}}(t_k)$.
If the Kalman gain is defined as:

$$\mathbf{K}(t_k) = \mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}}\left[\mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}}\mathbf{H}_{t_{k-1}} + \mathbf{R}_{t_{k-1}}\right]^{-1}, \tag{3.36}$$

and the above equations can be summarized to:

Algorithm 3.5.1 (Kalman Filter)**.** The optimal filter for the continuous-discrete system (3.24) and (3.25) consists of the conditional mean $\widehat{\mathbf{x}}(t_k)$ and conditional covariance $\mathbf{P}_{t_k}$. In between the observations the conditional mean and covariance satisfies (3.3),

$$\frac{d\widehat{\mathbf{x}}(t_k)}{dt} = \mathbf{F}_t\widehat{\mathbf{x}}(t_k) \tag{3.37}$$

$$\frac{d\mathbf{P}_{t_k}(t_k)}{dt} = \mathbf{F}_t\mathbf{P}_{t_k} + \mathbf{P}_{t_k}\mathbf{F}_t + \mathbf{G}_t\mathbf{Q}_t\mathbf{G}_t^T, \qquad t_k \le t < t_{k+1} \tag{3.38}$$

Given an observation at time point t_k, the conditional mean and covariance are updated using the following equations:

$$\widehat{\mathbf{x}}(t_k) = \widehat{\mathbf{x}}(t_{k-1}) + \mathbf{K}(t_k)\mathbf{d_k} \tag{3.39}$$

$$\mathbf{P}_{t_k} = \mathbf{P}_{t_{k-1}} + \mathbf{K}(t_k)\mathbf{H}_{t_{k-1}}\mathbf{P}_{t_{k-1}} \tag{3.40}$$

$$= \left[\mathbf{I} - \mathbf{K}(t_k)\mathbf{H}_{t_k}\right]\mathbf{P}_{t_{k-1}}\left[\mathbf{I} - \mathbf{K}(t_k)\mathbf{H}_{t_k}\right]^T + \mathbf{K}(t_k)\mathbf{R}_k\mathbf{K}^T(t_k), \tag{3.41}$$

where the Kalman gain $\mathbf{K}(t_k)$ is given by (3.36).

The equations of Theorem 3.5.1 are, of course, only valid in the linear case and could also be derived using the same assumptions described in (3.24) and (3.25) with moment equations (3.16) and (3.17).

3.6 Outlook on approximation filters

The moment equations (3.16) and (3.17) are of principal interest since it should be possible to solve the general filtering problem by calculating the proper moments. However, the only closed form solution of (3.16) and (3.17) is the Kalman Filter. The constraints on the Kalman Filter is that the state space model (3.1) has to be linear and have Gaussian densities. In many applications this is a reasonable assuption. However, as computer power has increased over the last decades nonlinear models have become more widely used and the filters therefore have to be adapted to accomodate the nonlinear models.
The model and observation operators $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ in (3.1) and (3.2) are in general nonlinear functions of the state, input and parameters. This makes it impossible to solve the (3.16) and (3.17) in closed form. The Extended Kalman Filter is an approximated solution to the nonlinear case where the model and observation operators $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ have been linearized by the means of Taylor expansions. The Extended Kalman Filter has proven a good approximation for state spaces models with weak nonlinearities. However, if the state space model is strongly nonlinear, then the Extended Kalman Filter approximation breaks down and the filter becomes unreliable.

Many approximation attempts have been made for (3.16) and (3.17) in in order to derive better filters than the Extended Kalman Filter to handle strong nonlinear models. Evaluating the (3.16) and (3.17) for higher order moments, e.g., including the Skewness and Kurtosis respectively, the third and fourth order moments can be included into the filter resulting in a more accurate approximation. By assuming that the conditional densities are nearly Gaussian the third and fourth order moments will nearly vanish. This filter is called the Gaussian second order filter [22]. Assuming that the conditional densities are symmetrical, almost all (>1) odd moments will vanish assuming that the conditional densities are concentrated close to the mean value, such that the fourth and higher order moments will vanish. Filters of this type are called truncate second order filters [22].
For the Gaussian sum filter [1], in the approximation of (3.16) and (3.17) it is assumed that the conditional density can be approximated by a weighted sum of Gaussian kernels. Non-Gaussian densities can, to some degree, be accurately approximated with a large number of Gaussian kernels to produce tractable solutions to (3.16) and (3.17). This approximation can be thought of as several extended Kalman Filters running in parallel in order to produce a suboptimal solution estimate [9].
Another way around the approximation of (3.16) and (3.17) is to use the derivative free approximation, the Unscented Transformation [23]. The idea is to deterministically sample a minimal number of sigma points around the mean. These points are propagated through the nonlinear model hoping that this will give a better representation of the mean and covariance. The main advantage of the Unscented Kalman Filter is that the Jacobian of model operators $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ does not have to be calculated [9].
The full list of suggested approximations of (3.16) and (3.17) is long. Only a few common versions of the approximation are stated here. However, in data assimilation many of the above filters are not tractable since the approximation requires finding the tangent linear model operator. The challenge of storing the covariance and Kalman gain matrices at every update point makes even the Extended Kalman Filter very computationally expensive [13].
In recent years, the Ensemble Kalman Filter (EnKF) [13, 19] has gained

popularity in the data assimilation community. The implementation of the EnKF is based around Monte Carlo integration of (3.16) and (3.17) to obtain a sample estimate of the mean and the covariance. The major advantage of this implementation of the Kalman Filter is that it is derivative free and does not require storage of the covariance and Kalman matrices at every update point. However, a drawback is that the Monte Carlo samples have to be made with the dynamic model which is also very computationally expensive. Another drawback is the Gaussian density assumption undergoning the method.
On the frontiers of data assimilation is particle filtering. Particle filtering is a solution to the general filtering problem posed by the Monte Carlo simulation of (3.15). Particle filtering seeks to integrate the entire conditional density without any assumptions on densities. All moments can then be calculated from the conditional distribution. However, experience shows that in order to obtain the proper statistics to calculate the moments, a huge number of particles must be generated. If the dimension of the state equation is large, then the particle filters have the tendency to collapse and give unreliable estimates. This is called the curse of dimensionality. The EnKF can be considered a special case among the particle filters, since the EnKF only takes the first two moments into account, thereby making the EnKF more robust as fewer particles are needed for a proper representation of the first two moments.

3.7 Monte Carlo based filtering

The basis of the particle filters and Ensemble Kalman Filters is the Monte Carlo integration. Monte Carlo simulation is the approximation that an arbitrary integral can be approximated with a finite sum of random samples from a probability distribution.
Assume that an arbitrary function exists $\mathbf{f}(\mathbf{x}) \in \mathcal{R}$ and the expectation can be calculated. Using the definition of the expectation by assuming that a probability distribution function (pdf) $p(\mathbf{x}) \in \mathcal{R}$ exists, the expectation can

be written as:

$$E[\mathbf{f}(\mathbf{x})] = \int_{-\infty}^{\infty} \mathbf{f}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \tag{3.42}$$

Most often the integral can not be solved analytically and therefore an approximation has to be obtained. A way around this is Monte Carlo integration.

3.7.1 Monte Carlo sampling

If it is assumed that it is possible to simulate N independent and identically distributed (i.i.d.) random samples from $\{\mathbf{x}^{(i)}; i = 1, \ldots, N\}$ according to the pdf $p(\mathbf{x}, \mathbf{y})$. As in the book by [12], the empirical distribution is defined as a sum of dirac delta functions,

$$p_N(d\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \delta_{\mathbf{x}^i}(d\mathbf{x}), \tag{3.43}$$

where $\delta_{\mathbf{x}^i}(d\mathbf{x})$ is the probability mass located in $\mathbf{x}^i$. The $d\mathbf{x}$ denotes that the exact location cannot be specified; the probability can only be specified in the vicinity of $\mathbf{x}^i$ by making a small sphere around $\mathbf{x}^i$. In accordance with the above definition the expectation (3.42) can then be estimated as:

$$\overline{E[\mathbf{f}(\mathbf{x})]} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{f}(\mathbf{x}^i), \tag{3.44}$$

where $\mathbf{f}$ is the considered function and, for generality, here made time dependent and multidimensional. According to law of large numbers, the expectation (3.44) will converge almost surely to (3.42), i.e. $\overline{E[\mathbf{f}(\mathbf{x})]} \xrightarrow[N\to\infty]{a.s.} E[\mathbf{f}(\mathbf{x})]$ and if the posterior variance of $\mathbf{f}(\mathbf{x})$ is bounded, i.e. $\sigma_{\mathbf{f}}^2 < \infty$ then the central limit theorem states, [39, 12]

$$\sqrt{N}\left(\overline{E[\mathbf{f}(\mathbf{x})]} - E[\mathbf{f}(\mathbf{x})]\right) \underset{N\to\infty}{\Longrightarrow} \mathcal{N}(0, \sigma_f^2),$$

where $\underset{N\to\infty}{\Longrightarrow}$ denotes convergence in distribution.

Any moment can be calculated from a finite number of samples from a random probability distribution evaluated through the nonlinear model $\mathbf{f}(\mathbf{x})$ with the residual sampling error $N^{-1/2}$. In the technical report in Appendix E, a discussion on how to sample efficiency from a posterior distribution $p(\mathbf{x}|\mathcal{F}_t)$ is explained in detail. On the basis of the derivation in the report, it is assumed that it is possible to sample sequentially from a posterior distribution by means of bootstrap sampling. Bootstrap sampling states that it is possible to estimate the prediction distribution $p(\mathbf{x}_{t+1}|\mathcal{F}_t)$ by sampling from the latest posterior conditional density $p(\mathbf{x}_{t_k}|\mathcal{F}_{t_k})$ through the nonlinear model $\mathbf{f}(\mathbf{x})$.

3.7.2 The Ensemble Kalman Filter

With the above definitions, the EnKF can be derived. In the orignal EnKF paper [13] the standard Kalman Filter equations were used where the empirical covariance matrix was substituted for the model covariance matrix. This, however, lead to spurious correlation in large scale models and also gave rise to filter divergence. The problem was later corrected in the paper [19]. The problem was attributed to the substitution of the empirical covariance matrix $\widehat{\mathbf{P}}$ into the normal covariance matrix $\mathbf{P}$. It was shown in [19] that EnkF reduced the total covariance too much and that for the observation covariance $\mathbf{KRK}^T$, the last term of (3.40) was missing in the standard Kalman Filter formulation with the adopted empirical covariance. Put in other terms: the state system is observed through the normal linear observation equation (this is the only linear assumption with EnKF), however, in the EnKF, the observations have to be treated as random samples from observation probability distribution,

$$\widehat{\mathbf{y}}_{t_k} = \mathbf{H}\widehat{\mathbf{x}}_{t_k} + \boldsymbol{\varepsilon}_{t_k}, \tag{3.45}$$

where $\boldsymbol{\varepsilon}_{t_k}$ is i.i.d. measurement error with zero mean and covariance $\mathbf{R}_{t_k}$. For the EnKF to work properly, it is necessary to treat the observations as draws from a distribution. The observations are treated as random variables that are drawn from the Gaussian density $\mathbf{y}^i_{t_k} \sim N(\widehat{\mathbf{y}}_{t_k}, \mathbf{R_k})$. It was shown

in the article that this did not violate any of the assumptions made on the original proposed EnKF. Treating the observations as random samples and with the definitions, the empirical conditional mean and covariance can be stated for the EnKF as:

Algorithm 3.7.1. Let the empirical mean and covariance be given by (3.44), then the ensemble Kalman Filter (EnKF) is given by the following equations:

$$\widehat{\mathbf{x}}^i_{t_k} = \mathbf{x}^i_{t_k} + \mathbf{K}_{t_k}\left(\mathbf{y}_{t_k} + \boldsymbol{\varepsilon}^i_{t_k} - \mathbf{H}_{t_k}\mathbf{x}^i_{t_k}\right) \tag{3.46}$$

$$\widehat{\mathbf{P}}_{t_k} = \left(\mathbf{I} - \mathbf{K}_{t_{k-1}}\mathbf{H}^T_{t_{k-1}}\right)\widehat{\mathbf{P}}_{t_{k-1}}, \tag{3.47}$$

where the residual sampling error of the $\widehat{\mathbf{P}}_{t_k}$ is of the order $N^{-1/2}$. The conditional mean is found by taking the expectation of (3.46).

3.7.3 The particle filters

In Appendix E, the discrete-discrete filtering problem is stated for the basic particle filters. The particle filters treated in the report are the Sequential Importance Sampling and Resampling filters, (SIS) and (SIR), respectively. There is also a section on particle smoothers, namely the backward particle smoother and the two filters smoother, respectively. However, the continuous-discrete filter is not mentioned in the report. Therefore an small introduction to the continuous-discrete bootstrap filter will be given here. Most of the preliminaries have already been covered in Section 3.7 on finite approximation by Monte Carlo integration of the moment generating equations (3.15). The algorithm for the continuous-discrete bootstrap filter with resampling will be stated without further details than already discussed in the report and the above sections. Assume that the state equation is given by a stochastic differential equation as in (3.1) and that the system is observed through $\mathbf{y}_k$, i.e.

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + \mathbf{G}(\mathbf{x}_t, t)d\mathbf{B}_t, \qquad t \geq t_0, \tag{3.48}$$

where $\mathbf{f}$ and $\mathbf{x}$ are p valued vectors and $\mathbf{G}$ is a $p \times q$ matrix, $\mathbf{B}_t$ is q valued Brownian motion with $E[d\mathbf{B}_t d\mathbf{B}_t^T] = \mathbf{Q}(t)dt$. The discrete observations are given as:

$$\mathbf{y}_k \sim p(\mathbf{y}_k|\mathbf{x_{t_k}}) \tag{3.49}$$

where $p(\mathbf{y}_k|\mathbf{x_{t_k}})$ is assumed to be Gaussian distributed measurement errors (3.8) and the conditional density can be expressed as in (3.9). Note that in the observation operator (3.49) there are no constraints on the observation operator $\mathbf{h}_k(\mathbf{x}_{t_k}, t)$.

Algorithm 3.7.2 (Continuous-discrete bootstrap filter with resampling)**.** The bootstrap filter for stochastic differential equation with discrete observation can be achieved by performing the following steps

i Simulate trajectories between observations $\{\mathbf{x}^i(t) : t_{k-1} \leq t \leq t_k, i = 1, N\}$ from the equation by numerical integration,

$$d\mathbf{x}_t^i = \mathbf{f}(\mathbf{x}_t^i, t)dt + \mathbf{G}(\mathbf{x}_t^i, t)d\mathbf{B}_t^i \tag{3.50}$$

$$\mathbf{x}^i(t_k) = \mathbf{x}_{k-1}^i \tag{3.51}$$

with independent Brownian motion $\mathbf{B}_t^i$ and set $\mathbf{x}_k^i = \mathbf{x}^i(t_k)$. Now each $\mathbf{x}^i(t_k)$ is a random draw from the transition distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$.

ii Compute new weights

$$\omega_k^i = \omega_{k-1}^i p(\mathbf{y}_k|\mathbf{x_k^i}) \tag{3.52}$$

iii Resample the particles with probability ω_k^i from the set $\{x_k^i, i = 1, \ldots, N\}$.

The definition of the importance weights ω_k^i can be found in Appendix E and in [20].

3.8 Discussion

3.8.1 Note on SMC filters

The bootstrap particle filter or SIR filter is a very simple algorithm that uses the dynamic model as the importance distribution to sample from. This is, however, not very efficient and can often lead to filter divergence if the dynamical model is not very accurate. Finding proper importance distributions to the dynamical models is an ongoing topic of research [14, 6, 11, 36, 35].
In geophysical data assimilation applications, the above approaches are too difficult since some of the approximations are too difficult, i.e. the Girsanov theorem is hard to implement in a large scale model. This makes it very difficult to get efficient importance distributions. This lack, in turn, leads to filter divergence because the importance weights can collapses and the prediction skill will be lost. This is known as the curse of dimensionality. In the paper [4], very detailed investigations on importance distributions are carried out. The conclusion of the paper states that in order to get efficient importance distributions, one must either sample from a proposal distribution that mimics the model in some way or apply some form of localization techniques to the importance distribution. In the first case, a surrogate model is constructed from an EOF analysis of a main model and acts as a proposal distribution to the importance distribution in the paper [5]. However, this did not solve the problem in the paper and further investigations have to be carried out. In the latter case, the importance distribution is modified in a local area to reduce the importance distribution to a set of local importance distributions.
In the paper [3], a modified EnKF filter with a Gaussian mixture [1] was applied to two test cases. In this approach, the localization step introduced discontinuities across the different localization areas and an additional EnKF steered smoothing step was needed. In the paper [16], which can be found in the Appendix A, a similar approach was tested with a hybrid particle filter setup. The particle filter uses a local importance distribution step in order to reduce the dimension of the importance distribution in the

local area with a resampling step. In the global area, a correctness step was introduced in order to cancel out the effect of having used the observation twice. The hybrid particle filter was successfully tested for a series of different dimensions on a scalable stochastic differential equation. The test showed that with the introduction of a localization step, the prediction of the hybrid particle filter matched that of the EnKF. The EnKF filter does not, to the same degree, suffer under the curse of dimensionality as the particle filters on intermediate numerical models. However, the curse of dimensionality is evident in very large systems and various approaches have also been used with success. In the introductory section of paper [16], a brief introduction to the localization techniques is given.
Bootstrap particle filters are very easy to use for any system. However, most often it is necessary to modify the particle filters in order to get good filter performance. The book [12], is devoted to particle filters and how to use them efficiently on many different model settings. Note that the particle filters described in [12] are all of the discrete-discrete filter types. However, many of the particle filter applications can easily be modified for use in a continuous-discrete filter setting.

3.8.2 Simple implementation of Optimal Interpolation filter

In the paper C, an Optimal Interpolation filter was used in a simple data assimilation experiment carried out on an atmospheric chemical transport model (CTM). The Optimal Interpolation (OI) filter [17, 10, 24] can be interpreted as an ordinary Kalman Filter without any time evolution of the covariance matrices. The data assimilation task was to investigate the benefited of data assimilation in a CTM. The Ordinary Kalman Filter was consider for the experiment, however, it was found too computationally resource-demanding because of the storage of the covariance matrices upon each observation update. The EnKF was considered *black magic* at the time of the experiment and hence the choice was made to employ the Optimal Interpolation filter. The OI filter had been used at ECMWF [28] up until 1996 in their global weather forecast models. It was shown that the OI filter

was a robust filter that gave good results in spatial-temporal domains. As a preliminary filter, it was very effective in the reducing RMSE and bias in the model predictions. As noted, it was later revealed that there is an equivalent SMC interpretation of the OI filter called the steady-state EnKF or the EnOI filter. The steady state EnKF has been used extensively at DHI.

3.8.3 Surrogate data assimilation

In the paper D, some experiments with a surrogate model are documented. Data assimilation simulations can often take days to complete due to the heavy computations involved. To reduce the complexity of the models and thereby the computer time, simpler models can be constructed based on the full models. Simpler model should, however, still posses some of the characteristics of the full model in order to serve as a legitimate alternative to the full model. Surrogate models can take on lot of different shapes. In the paper B, a simpler model was constructed from the principal equations where some of the complexities were left out. However, the approach in the paper D, was quite different. The model used in the experiment was a 2 dimensional barotropic model, MIKE 21. The surrogate modelling technique applied is based on a first order Taylor Series expansion of this barotropic model in a reduced space spanned by covariance eigenvectors derived from an empirical orthogonal function analysis. The experiment demonstrates that a significantly smaller computational effort can provide uncertainty estimates that resemble Monte Carlo estimates using the high-dimensional complex model. The uncertainty estimates conducted in the experiment could be used as a proposal distribution to the barotropic model in a real data assimilation setting.

Part IV

Likelihood Inference

CHAPTER 4

Likelihood Inference

4.1 Maximum likelihood method

Maximum likelihood methods for estimating parameters embedded in stochastic differential equations are treated in the paper [15] in Appendix B. In this paper, a set of ordinary differential equations (ODE) was augmented to stochastic differential equations by adding a diffusion term with constant diffusion coefficient. Allowing the ODEs to be augmented into SDEs gave rise to the prediction error decomposition method where model and observation errors are identified as two different processes. This has an obvious advantage over the normal output error method used in typical geophysical modeling.
The advantages which can be listed for a continuous-discrete stochastic state space formulation (3.1) - (3.2) originate in part from the fact that the diffusion part of the SDE accounts for: [29]

- *Modeling approximations.* For instance, the dynamics, as described by the model operator $\mathbf{f}$ in (3.1), might be an approximation of the true system.

- *Unrecognized and unmodeled inputs.* Some variables which are not considered.
- *Measurements of the input are noise-corrupted.* In this case, the measured input is regarded as the actual input to the system, and the deviation from the true input contributes to $d\mathbf{B}_t$

The maximum likelihood method will facilitate an estimation of every parameter in the system, including parameter describing the model noise, input noise and observation noise. However, before the Maximum likelihood is outlined, there are still some properties that have to be explored. The main result of the Fokker-Planck equation is that the density for the model state distribution is given as a transition distribution from the previous time step to current time step multiplied by previous probability distribution. More formally,

$$p(\mathbf{x}_t, \mathcal{F}_{k-1}; \boldsymbol{\theta}) = \int p(\mathbf{x}_t|\mathbf{x}_{t_{k-1}}; \boldsymbol{\theta}) p(\mathbf{x}_{t-1}, \mathcal{F}_{k-1}; \boldsymbol{\theta}) d\mathbf{x}_{t_{k-1}} \tag{4.1}$$

If all the transition probability distributions $p(\mathbf{x}_{t_k}|\mathbf{x}_{t_{k-1}}; \boldsymbol{\theta})$ for a given time series are available then the maximum likelihood function becomes,

$$\mathbf{L}(\boldsymbol{\theta}) = \prod_{k=1}^{K} p(\mathbf{x}_{t_k}|\mathbf{x}_{t_{k-1}}; \boldsymbol{\theta}) \tag{4.2}$$

However, the observations are not directly observable through $\mathbf{x}_{\mathbf{t_k}}$. The observations are only given as noisy discrete measurements which are functions of the state. The maximum likelihood function can then be evaluated using a filter. Expressing the maximum likelihood through the observations, the following relationship can be derived,

$$L(\boldsymbol{\theta}; \mathcal{Y}_K) = p(\mathcal{Y}_K|\boldsymbol{\theta}), \tag{4.3}$$

where $\mathcal{Y}_K$ is a sequence of measurements time $k \in \{1, \ldots, K\}$, $\boldsymbol{\theta}$ is the argument that maximizes the likelihood function. The likelihood function is defined as the joint probability density for all the observations. Using the

Bayesian framework $P(A \cap B) = P(A|B)P(B)$, the likelihood function can be rewritten as products of conditional probability densities:

$$\begin{aligned} L(\boldsymbol{\theta}; \mathcal{Y}_K) &= \left(\prod_{t=1}^{K} p(\mathbf{y}_{t_k} | \mathcal{Y}_{t_{k-1}}, \boldsymbol{\theta}) \right) p(\mathbf{y}_0 | \boldsymbol{\theta}) \\ &= \prod_{k=1}^{K} \left(\int p(\mathbf{y}_{t_k} | \mathbf{x}_{t_k}, \boldsymbol{\theta}) p(\mathbf{x}_t | \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) d\mathbf{x}_{t_k} \right) p(\mathbf{y}_0 | \boldsymbol{\theta}), \end{aligned} \tag{4.4}$$

where $p(\mathbf{y}_k | \mathbf{x}_{t_k}, \boldsymbol{\theta})$ is probability density given $\mathbf{x}_{t_k}$. Hereby the relationship between the observations and the state space model is established. It can be put into a more practical setting by introducing the one-step prediction, the measurement covariance and the prediction error,

$$\begin{aligned} \hat{\mathbf{y}}_{k|k-1} &= E\{\mathbf{y}_k | \mathcal{Y}_{t_{k-1}}, \boldsymbol{\theta}\} \\ \mathbf{R}_{k|k-1} &= V\{\mathbf{y}_k | \mathcal{Y}_{k-1}, \boldsymbol{\theta}\} \\ \boldsymbol{\varepsilon}_{t_k} &= \mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1} \end{aligned} \tag{4.5}$$

and assuming that the conditional probability densities are Gaussian (4.4) can be rewritten as:

$$L(\boldsymbol{\theta}; \mathcal{Y}_K) = \left(\prod_{k=1}^{K} \frac{\exp\left(-\frac{1}{2} \boldsymbol{\varepsilon}_k^K \mathbf{R}_{k|k-1}^{-1} \boldsymbol{\varepsilon}_{t_k}\right)}{\sqrt{\det\left(\mathbf{R}_{k|k-1}\right)} \sqrt{2\pi}^n} \right) p(\mathbf{y}_0 | \boldsymbol{\theta}),$$

where $\mathbf{R}_{k|k-1}$ and $\boldsymbol{\varepsilon}_{t_k}$ can be evaluated using the filter for non-linear models, and where n denotes the number of outputs in the vector $\mathbf{y}_k$.

4.2 Maximum Likelihood via SMC methods

The estimation of model parameters is a very important part of filtering. However, it is not a straight forward matter with the SMC filters. Ordinary methods fail very often because of the noisiness of the SMC filters. There is simply too much noise in the system for normal maximum likelihood methods. Usually, it is no problem to find the neighbourhood of the

maximum likelihood; however, the problem is to find the exact maximum of the likelihood function. If only a limited number of particles are used, then the likelihood function will not be smooth enough and rather noisy. With such a noisy likelihood function the ordinary search methods will get stuck in local maxima around the global maximum. A way to obtain a smoother likelihood function is to fix the seed in the random number generator. However, this hack will ensure the same likelihood function at the optimization iteration step and, therefore, some of the local maxima may be smoothened. However, it will not guarantee the maximum likelihood estimate.
There have been many examples of methods that reduce the noise in the system parameters such that the optimization converges to the maximum likelihood estimate. Maximum likelihood estimates (MLE) for SMC methods, e.g. particle filters and the ensemble Kalman Filter, are hard to obtain if the likelihood function cannot be written in closed form. For a closed form likelihood function there is the universal method of the Monte Carlo Expectation Maximization (MCEM) [40]. This method will almost always give the maximum likelihood estimate. The maximization of the MCEM algorithm comes with the cost of incorporating a smoother step, which will make the MCEM very resource demanding. A cheaper smoother method for some discrete and continuous models has be proposed [32]. This method is good for models with known likelihood functions. Contrary to the traditional particle smoothing methods described in Appendix E, this smoother is much faster since it only considers particles within an L-step range from where the smooth particles are wanted.
Attempts have been made to make the MCEM algorithm applicable to non-closed form likelihood functions [37]. However, the likelihood in these attempts are still too noisy to converge to the MLE. Looking to gradient based search methods e.g. Stochastic Approximation techniques (SA) and Simultaneous Perturbation Stochastic Approximation (SPSA) [38], in the most basic setting the MLE is found by calculating the score function from two different positions and then calculating the gradient. By iteration, the gradient will go to zero as the parameters go towards the MLE. For SMC the SA methods will only converge towards the MLE, however, it will not find the MLE since the score function is too noisy in most cases for the

gradient to converge to the MLE. This has been shown by [8] that a simple nonlinear model (same benchmark model as used in Appendix E) with two parameters would not converge after 1,400,000 iterations. This is mainly due to the noise in the objective function. In another paper the SPSA algorithm was used on the same model where three model parameters were to be estimated. The authors had better success with the SPSA, however, it is still evident from the paper that there is still much noise in the objective function. An important fact that the authors stressed, special care has to be taken when choosing the cooling parameter for the SPSA algorithm. The above discussion leads to the next section where an adaptive stochastic approximation techniques is introduced.

4.2.1 Maximum Likelihood via Iterated Filtering

A new gradient search method suggested by [21] takes into account that the likelihood function is noisy around the maximum. The paper suggests that reducing the variance at each iteration step will ensure that the maximum is reached. The basic idea of the procedure is to replace the original model with a model that has close receplance. The time-constant parameter θ is substituted with a time-varying process θ_t. The densities $f(x_t|x_{t-1},\theta)$, $f(y_t|x_t,\theta)$, and $f(x_0|\theta)$ of the time constant model are substitutted with $f(x_t|x_{t-1},\theta_t)$, $f(y_t|x_t,\theta_t)$, and $f(x_0|\theta_t)$. The time-varing process θ_t is taken to be a random walk in $\mathcal{R}^{d_\theta}$. The main algorithm only depends on the mean and varaince of the random walk, which are defined as,

$$E[\theta_t|\theta_{t-1}] = \theta_{t-1} \qquad Var(\theta_t|\theta_{t-1}) = \sigma^2\Sigma \tag{4.6}$$

$$E[\theta_0] = \theta \qquad Var(\theta_0) = c^2\sigma^2\Sigma, \tag{4.7}$$

where σ and c are scalar quantities and the new model is idential to the non-time-varing parameter model by setting $\sigma = 0$. The idea is to obtain an estimate of θ by taking the limit $\sigma \to 0$. Σ is arbritary positive-definite symmetric matrix. It typicals holds the values of θ in the diagonal.

Algorithm 4.2.1 (Maximum Likelihood via Iterated Filtering)**.**

The following quantaties can be obtained from the SMC filter,

$$\begin{aligned} \widehat{\theta}_t &= \widehat{\theta}_t(\theta, \sigma) = E[\theta_t | y_{1:t}] \\ V_t &= V_t(\theta, \sigma) = Var(\theta_t | y_{1:t-1}) \end{aligned} \tag{4.8}$$

1. Chose stating values $\widehat{\theta}^{(1)}$, the discount factor $0 < \alpha < 1$, the intial variance multiplier c^2 and the number of iterations N
2. Do for $1, \ldots, N$
   i Set $\sigma_n = \alpha^{n-1}$. For $t = 1, \ldots, T$ evaluate $\widehat{\theta}_t^{(n)} = \widehat{\theta}_t(\widehat{\theta}^{(n)}, \sigma_n)$ and $V_{t,n} = V_t(\widehat{\theta}^{(n)}, \sigma_n)$
   ii Set $\widehat{\theta}^{(n+1)} = \widehat{\theta}^{(n)} + V_{1,n} \Sigma_{t=1}^{T} V_{t,n}^{-1} (\widehat{\theta}_t - \widehat{\theta}_{t-1})$
3. Take $\widehat{\theta}^{(N+1)}$ to the maximum likelihood estimate of the parameter θ for the fixed model.

The algorithm states that the parameters will be updated in the direction of the increasing local likelihood. The iteration will stop at the local maximum likelihood fix point. For more details of the MIF theorem, see [21] and the supporting text to the paper on the author's homepage.

4.3 Discussion

In paper B, a set of ordinary differential equations for a simplified ecosystem was augmented into stochastic differential equations. This was done to allow the maximum likelihood method to estimate the model parameters of the simplified ecosystem. As already mentioned above, the stochastic differential equation approach is superior to the output error method normally used in data assimilation. In the paper B, two geochemical components were measured on a monthly basis throughout 2001 for a fresh water lake in northern Zealand, Denmark. Input data was collect from sites nearby. Measurement campaigns are quite expensive which was evident from the measured data set. The geochemical components were only measured once

monthly due to the high costs involved in the measuring campaign. The limited data set was used to estimate the model parameters in the simplified ecosystem model. Due to the small data set, the parameters had a high standard deviation connected to the estimation. The uncertainties could have been reduced if there had been more data. However, this was not the case. In the near future, new, inexpensive measurement buoys will be able to log measurements at a high frequency. When this type of measurement equipment becomes available, the method presented in paper B will be ideal for the task.

Since data was very limited, the maximum likelihood via iterated filtering was considered overkill as an estimator for the simplified model. The message of using stochastic differential equation was deemed more important than estimating the parameters from a SMC filter perspective. If the inference had been conducted with the MIF algorithm, then the emphasis would have been on the SMC filter and not on the stochastic differential equation approach. This message was paramount to communicate to the data assimilation community. The estimator *CTSM* in paper B had also been used in many other applications and was well documented in international journals for its reliability and robustness.

However, some preliminary results were obtained with the MIF. To test the implementation of the MIF in the parallel FORTRAN program, a small test case was chosen. The Ornstein-Uhlenbeck process was selected as the test case since it is one of the few stochastic differential equations with close form solution. The Ornstein-Uhlenbeck process is defined as:

$$dX_t = -AX_t dt + \sigma dB_t \tag{4.9}$$

$$Y_{t_k} = X_{t_k} + \varepsilon_{t_k}, \tag{4.10}$$

where $A > 0$ is the drift coefficient, $\sigma > 0$ is the diffusion coefficient and B_t is a Brownian motion. The system is observed through Y_k with measurement noise ε_k. The conditional distribution of the latent state can be found through the Fokker-Planck equation and is given by:

$$p(x_{t+dt}|x_t) = \phi\left(x_{t+dt}; e^{-Adt}x_t; \frac{\sigma^2}{2A}\left(1 - e^{-2Adt}\right)\right). \tag{4.11}$$

Knowing the conditional transition densities, the ordinary Kalman Filter can be applied to calculate the maximum likelihood function using (4.6). The system 4.9 was simulated with $(A, \sigma, \Sigma) = (0.5, 0.8, 1)$. The Ornstein-Uhlenbeck process was discretized with the Euler-Maruyama approximation with time step $dt = 0.1$. The simulation to time $T = 100$, i.e. 1000 time steps and white noise with standard deviation $\Sigma = 1$, was added to the solution.
A simplex search algorithm was used together with the Kalman Filter to find the maximum likelihood values as the benchmark values for the MIF algorithm.
The results from the MIF via a bootstrapping particle filter can be seen in Figure 4.3. The MIF algorithm proved to be a challenge to implement. There where many small details that had to be solved in order to get good performance. In the above setup, 5000 particles were used and the seed had to be reset at each new iteration of the algorithm. Otherwise, the estimation would fail. However, once all the small details were out of the way the MIF algorithm proved to be quite robust and gave a good estimation results that matched the parameter values from the exact estimation. This is quite remarkable for a Monte Carlo method which normally has difficulties in reaching the right maximum in the likelihood function.
For large scale applications, the MIF algorithm is still too resource demanding even for today's computational standards. However, on smaller scales, such as the ecosystem in paper B, the MIF might be a robust estimator.

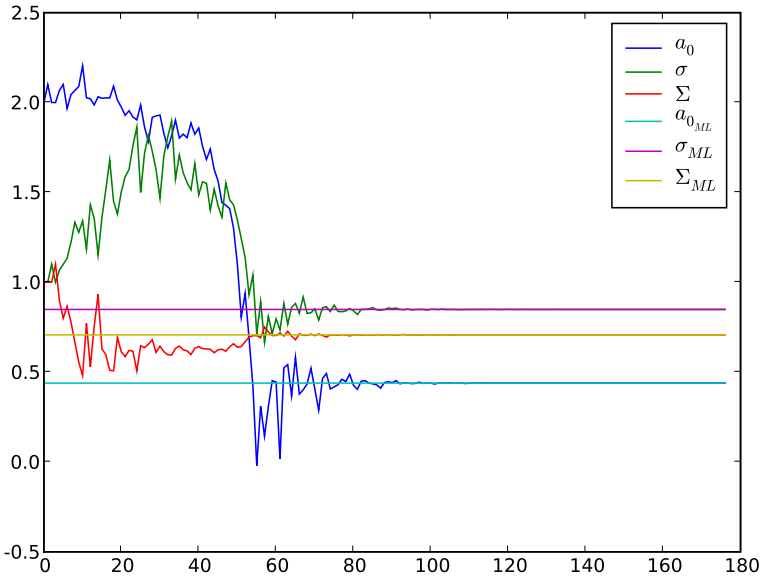


Figure 4.1: The subscript ML in the legend indicates that the parameter values are found through the Maximum likelihood method using the Kalman Filter. The parameters without subscript in legend is the MIF parameter estimates

Bibliography

[1] D. Alspach and H. Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972. ISSN 00189286.

[2] V.I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer Verlag, second edition, 1989. ISBN 0-387-96890-3.

[3] T. Bengtsson, C. Snyder, and D. Nychka. Toward a nonlinear ensemble filter for high-dimensional systems. *Journal of Geophysical Research*, 108(D24):STS2–1–10, 2003. ISSN 01480227.

[4] Thomas Bengtsson, Peter Bickel, and Bo Li. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. *IMS Collections - Probability and Statistics in Honor of David A. Freedman*, 2:316–334, 2008. doi: 10.1214/193940307000000518.

[5] L. Mark Berliner and Christopher Wikle. Approximate importance sampling Monte Carlo for data assimilation. *Phys. D*, 230(1-2):37–49, 2007. ISSN 0167-2789.

[6] Alexandros Beskos, Omiros Papaspiliopoulos, Gareth O. Roberts, and Paul Fearnhead. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):333–382, 2006. ISSN 13697412.

[7] Tomas Björk. *Arbitrage theory in continuous time*. Oxford Univ. Press, 2. ed., reprint. edition, 2005. ISBN 0-19-927126-7. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+505893878&sourceid=fbw_bibsonomy.

[8] Bao Ling Chan, Vladislav B. Tadic, and Arnaud Doucet. Optimisation of particle filters using simultaneous perturbation stochastic approximation. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 6:681–684, 2003. ISSN 07367791.

[9] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. Technical report, Communications Research Laboratory, McMaster University, Hamilton ON, Canada, 2003.

[10] R. Daley. *Atmospheric Data Analysis*. Cambridge University Press, 1996.

[11] Petros Dellaportas, Nial Friel, and Gareth O. Roberts. Bayesian model selection for partially observed diffusion models. *Biometrika*, 93(4):809–825, 2006. ISSN 00063444.

[12] A. Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, first edition, 2001. ISBN 0-387-95146-6.

[13] Geir Evensen. Sequential data assimilation with a nonlinear quasigeostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99:10143–10162, 1994.

[14] Paul Fearnhead, Omiros Papaspiliopoulos, and Gareth O. Roberts. Particle filters for partially observed diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):755–777, 2008. ISSN 13697412.

[15] Jan Frydendall, A. Eriksen, J. V. Tornfeldt, and H. Madsen. Calibration and estimation of models of a complex eco system. *Journal of Hydrology*, in preparation, 2009.

[16] Jan Frydendall, J. V. Tornfeldt, and H. Madsen. Importance sampling localization for sequential importance re-sampling filters. *Nonlinear Processes in Geophysics*, in review, 2009.

[17] L S Gandin. Objective analysis of meteorological fields. In *Program Scientific Translalions*, 1963.

[18] C.W. Gardiner. *Handbook of Stochastic Methods*. Springer Verlag, third edition, 2004. ISBN 3-540-20882-8.

[19] P. J. van Leeuwen Gerrit Burges and Geir Evensen. Analysis scheme in the ensemble kalman filter. *American Meteorological Society*, 126:1719–1724, 1998.

[20] John Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–1339, 1989.

[21] E. L. Ionides, C. Breto, and A. A. King. Inference for nonlinear dynamical systems. *PNAS*, 103(49):18438–18443, 2006. URL http://www.pnas.org/cgi/content/abstract/103/49/18438.

[22] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Dover Publications INC, first edition, 1970. ISBN 13: 978-0-48646274-5.

[23] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. ISSN 00189219.

[24] E. Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press, 2003.

[25] Ioannis Karatzas and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer Verlag, second edition, 1991. ISBN 3-387-97655-8.

[26] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer Verlag, thrid edition, 1999. ISBN 3-540-54062-8.

[27] Harald E. Krogstad. The kolmogorov and fokker-planck equations. Technical report, Department of Mathematical Sciences, Institutt for matematiske fag, NTNU, N-7491 TRONDHEIM, 2003.

[28] A. C. Lorenc. A global 3-dimensional multivariate statistical interpolation scheme. *Monthly Weather Review*, 109(4):701–721, 1981.

[29] H. Madsen and J. Holst. Estimation of continuous-time models for the heat dynamics of a building. *Energy and Building*, 22:67–79, 1995.

[30] Henrik Madsen. *Time series analysis*. Chapman & Hall/CRC, 2007.

[31] Bernt Øksendal. *Stochastic Differential Equations*. Springer Verlag, 6th edition, 2005. ISBN 3-540-25662-8.

[32] Jimmy Olsson, Olivier Cappe, Randal Douc, and Eric Moulines. Sequential monte carlo smoothing with application to parameter estimation in non-linear state space models. *BERNOULLI*, 14:155, 2008. URL doi:10.3150/07-BEJ6150.

[33] Martin Wæver Pedersen, David Righton, Uffe Høgsbro Thygesen, Ken Haste Andersen, and Henrik Madsen. Geolocation of north sea cod (gadus morhua) using hidden markov models and behavioural switching. *Canadian Journal of Fisheries and Aquatic Sciences*, 65(11): 2367–2377, 2008. ISSN 0706-652X.

[34] K. B. Petersen and M. S. Pedersen. The matrix cookbook, oct 2008. URL http://www2.imm.dtu.dk/pubdb/p.php?3274. Version 20081110.

[35] Nilanjan Saha and D. Roy. A girsanov particle filter in nonlinear engineering dynamics. *Physics Letters A*, 373(6):627–635, 2009. ISSN 03759601.

[36] Simo Särkkä and Tommi Sottinen. Application of girsanov theorem to particle filtering of discretely observed continuous-time non-linear systems. *Bayesian Analysis*, 3:555–584, 2008.

[37] T.B. Schon, A. Wills, and B. Ninness. Maximum likelihood nonlinear system estimation. *IFAC Symposium Modelling, Identification and Signal Processing (SYSID 2006) Identification and System Parameter Estimation*, page 6 pp., 2006. ISSN noissn1188289620.

[38] James C. Spall. Stochastic optimization and the simultaneous perturbation method. *Winter Simulation Conference Proceedings*, 1:101–109, 1999. ISSN 02750708.

[39] Rudolph van der Merwe, Arnaund Doucent, Nando de Freitas, and Eric Wan. The unscented particle filter. Technical report, Cambridge University Engineering Department, 2000.

[40] Greg C. G. Wei and Martin A. Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704 and 2290005, 1990. ISSN 01621459.

Appendices

APPENDIX A

Paper A

Submitted to Physica D

Importance sampling localization for Sequential Importance Re-sampling filters

Jan Frydendall[a,b], E. Lindström[c], J.V.Tornfeldt Sørensen[b], H. Madsen[a]

[a] *DTU Informatatics, Richard Petersens Plads,DTU,Building 321, DK-2800, Lyngby, Denmark*
[b] *DHI, Agern AllÉ 5, DK-2970, Hørsholm, Denmark*
[c] *Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sölvegatan 18, Lund, Sweden*

Abstract

State of the art for nonlinear geophysical data assimilation in large state space models is the Ensemble Kalman Filter (EnKF). Significant contributions to the field in various applications have been made for smaller state spaces by the introduction of particle filters (PF). These contributions have all had to deal with the curse of dimensionality which particularly the PFs suffer from. The PF has found its successful application in small state space models of the order 10^1 for systems where strong nonlinear behavior with fast dynamics are encountered. Extending the size of the state space from 10^1 to 10^2 will in such systems deteriorate the predictability skill of the PFs and the importance weights will collapse. In this paper a new method is suggested. Localization of the importance sampling domain greatly improves the predictability of the PFs for increasing state space size. The method is demonstrated in the Lorenz system of 1995 which is a perfectly scalable and strongly nonlinear system. The predictability of the PFs is first shown to match the EnKF for small state spaces in order to benchmark the PFs for the considered model. Further, the predictability of the PFs is investigated with and without the new localization method for several state space sizes of the Lorenz system. The importance sampling localization clearly extends the predictability of the PF to larger state spaces.

Keywords: Particle Filters, Ensemble Kalman Filter, Localization

1. Introduction

Non-linear filtering is one of the frontiers in geophysical data assimilation. At present, a corner stone of non-linear filtering is the Ensemble Kalman Filter (EnKF) [9, 5]. The EnKF is the preferred choice for many oceanographers and meteorologists. In the large state space dimensions of the order 10^4-10^7 encountered in these disciplines, the EnKF has proven to be a robust estimator.

The EnKF has many nice features which are not supported by the traditional Kalman filter and the family of variational analysis schemes [22]. The EnKF can fully handle non-linear propagation of the states, whereas the variational and extended Kalman filters have to partially linearize the model in order to make propagations of the states and its uncertainty. It can also handle non-linear measurement operators to some degree. The update step of the EnKF is a linear update, since the Kalman gain only use of the (co)-variance and average of the ensemble members representing the model states. If the measurements are strongly non-linear functions of the model state then the linear matrix formulation of the estimator breaks down. However, if the measurements are weakly non-linear functions of the model state then the assimilation scheme can use them successfully [10]. These restrictions do not pose a limitation in many applications and the EnKF delivers robustness and good results. The robustness is further strengthened when applying regularization to suppress spurious correlations [23].

Thus the EnKF has manifested itself as a robust technique for large state space models. However, efforts have be made to address its limitations through the introduction of the particle filter (PF) into geophysical data assimilation. PFs are fully capable of handling any non-linearity in the model and in the measurement operators. This together with the fact that the state update is perfectly consistent with the model dynamics, makes the PF very attractive. Early attempts implementing a PF into a oceanographic model turned out to be infeasible do to the number of particles required to keep the particles weights from collapsing into one [26]. In a later development a resampling step was introduced to avoid singularity of the particle weights. This reduced the required particle size dramatically [25]. However, this did not solve the intrinsic curse of dimensionality problems of the PF for large state space models, where a very large number of particles is needed for a proper update of the states.

PFs have been used with success in smaller state space models. Important examples of models with smaller state spaces are marine and lake ecosystem models. These models have state space dimensions of the order 10^1 - 10^2 and are thus much better suited for the PFs. These smaller models typically have strongly non-linear dynamics dominating in the propagation of the system. Ecosystem models addressed by the PF so far have typical been 0-D with only a few states e.g. components. The model used by [20, 19] is 0-d with four states model. This work of [20, 19] has demonstrated that the PF can be successfully applied to smaller state space ecosystem models. However, 1-D ecosystem models are typically divided into 10 to 15 layers in the vertical in order to include a good representation of the pycnocline. If just a simple ecosystem model with e.g. 4 states variables is discretized in a 1-D model, the state space will have

a size of 40 to 60 and this number will rapidly grow as more ecosystem processes and denser discretization are added. For such intermediate size state space models the applicably of the PF depends of the model dynamic and the data set assimilated, but its limitation in terms of number of particles required will soon be meet. However, just applying brute force to the particle filter singularity problem is not a solution as proved by [3]. A solution to the problem is to reduce the effective dimensionality of the state space. One approach to this is to construct proposal distribution from a lower dimensional model that to some extent mimics the model dynamics. This has been done by [4] where an smaller model is constructed from the original model by performing an empirical orthogonal functions of the model. The result of the dimensional reduction were promising, however, it did not solve the problem completely. Another approach is to exploit localization concepts.

An important development in the development of the EnKF towards robust applications has been the introduction of localization in the updating step [14, 13]. Spurious correlation becomes important when observability is low and model uncertainty high. This affects the Kalman gain, but it can be avoided by assuming spatially localized impact of each measurement and the robustness and skill of the EnKF is hence increased. Unfortunately such distance regularization is not applicable in the same formulation to the PFs, where correlations are not directly considered.

In this paper a method is suggested to reduce the dimensionality of the importance sampling distribution by introducing the concept of localization in the importance weights. The method suggested in this paper is more related to the distance regularization schemes of the EnKF. For a state space size of the 10^1 the particle filters are comparable with the EnKF. However, when increasing the state space dimension to the order of 10^2 the particle filters will deteriorate if the particle size is maintained. The method in this paper suggest that the state space can be decomposed into smaller domains, e.g. local regions of the state space. In these local regions a local resampling can improve the prediction performance of the particle filters.

The paper will compare the predictability of the PF and the EnKF for the non-linear Lorenz system (L95). The predictability of the filters is investigated through a) a dynamical system with small state space that will give motivation for solving the problems encountered in b) a dynamical system with intermediate size state space, by exploiting the effects of localization of the importance weights.

The L95 system is described in section 2. The theory of the localized resampling will be elaborated in the theory section 3.The experiments consist of two different set ups on the test bed and are described in section 4. Section 5 summarizes the paper and discusses the performance of the localizations.

2. The Lorenz system

The Lorenz system L95 [17, 18] is a simple equation that is supposed to mimic a simple atmosphere over the hemisphere with oceans and land masses on a sparse data assimilation network. In the this paper it will be used in a completely different setting. The L95 is augmented to a stochastic differential equation (SDE) by adding a diffusion term to the normal ordinary differential equation (ODE). It will denoted sL95 to differentiate it from the ordinary L95.

$$dx_i = [x_{i-1}x_{i+1} - x_{i-2}x_{i-1} - x_i]dt + Fdt + \sigma dB_i, \qquad (1)$$

where (x_i, F, σ, dB) is the state at the i^{th} node, the forcing parameter, diffusion coefficient and increments of the Brownian motion respectively. The equations are solved on 8 nodes using periodic boundary conditions. A figure of the setup is shown in Figure (5), however, in this figure there are 48 nodes.

In this setup the forcing parameter and diffusion coefficient are chosen to be $(F, \sigma) = (10, 4)$. Since the diffusion coefficient is a constant the Stratonovich and the Itô interpretations coincide and a normal integration scheme can be used. Ideally should the SDE be integrated with stochastic Runge-Kutta algorithm [6]. However, experience has shown that a normal Runge-Kutta scheme can be used as long as the diffusive term is smaller than the dynamics of the ODE. In this paper the diffusive term is sufficiently small such that a normal Runge-Kutta of order 4 can be implemented. A simulation from the sL95 can be seen in Figure (1). The dynamical variability of the sL95 is clearly much larger than the underlying diffusive process.

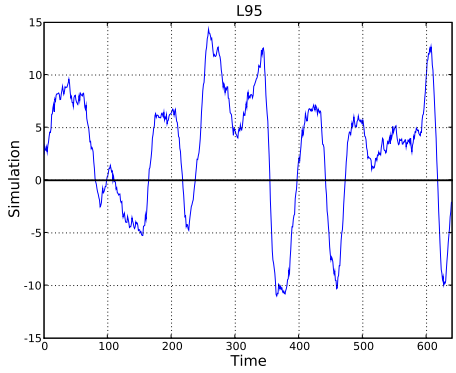


Figure 1: Simulation of the sL95. The dynamics of sL95 is much dominating then the underlying the diffusive process.

3. Theory

The EnKF has been described thoroughly in literature and will only be given a small introduction in this paper. The EnKF will be used in its original form [9, 5].

The dynamical system described in (1) is a 1^{st} order Markov process and therefore it can be put on state space form. The stochastic state space model is made up from two equations: the system equation for the evolution of the states and the observation equations relating the noisy measurements to the states. The systems equation is given as:

$$\mathbf{x}_{t+1} = \mathbf{f}_t(\mathbf{x}_t, \mathbf{w}_t), \tag{2}$$

where $\mathbf{f}$ is model transition function, $\mathbf{w}_t$ is normal distributed white noise with a known distribution for all t. When the discrete measurements $\mathbf{y}_t$ becomes available they are related to the system states via the observation equation:

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t, \mathbf{v}_t), \tag{3}$$

where $\mathbf{h}_t$ is the observation operator and $\mathbf{v}_t$ is another normal distributed white noise process with known distribution up to time t. The probabilistic form of the left hand side of equations (2) and (3) can be written as $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$ respectively.

The Bayesian formulation of the filter problem is to obtain the pdf of $\mathbf{x}_t$ from the information gathered up to time t, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. Based on the assumption that the state space is a 1^{st} order Markov process, the pdf can be estimated recursively. The recursive estimation of the pdf has to be done in two steps, a prediction step

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\mathbf{d}\mathbf{x}_{t-1} \tag{4}$$

and a measurement update step

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{\int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t\mathbf{y}_{1:t-1})\mathbf{d}\mathbf{x}_t}. \tag{5}$$

For the special case of linear model and Gaussian noise, $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ will remain Gaussian after every update of the filter [11]. The above equations can only be solved analytically if the pdfs are Gaussian, i.e. for a linear model and Gaussian noise or for a finite state space as this would transform the integrals into sums. The traditional Kalman Filter [15] is a linear filter solution which reduces the problem to propagating and updating the mean and covariance of the distribution. Another way of solving equations (4) and (5), which handles non-Gaussian pdfs, is to perform Monte Carlo sampling of the pdfs. The Monte Carlo approach will be able to describe the pdfs accurately if sufficiently many samples are used. The EnKF is a Monte Carlo sampling filter that in short uses Monte Carlo sampling to generate the propagation pdf in equation (4). The EnKF is capable of propagating non-Gaussian distributions, but the EnKF is still limited to the linear updates. The updated pdfs will only be generated from the first and second order moments through the Best Linear Unbiased Estimator (BLUE) as expressed in the following equations, [9]:

$$\mathbf{x}^u = \mathbf{x}^p + \mathbf{K}(\mathbf{y} - \mathbf{H}\mathbf{x}^p + \boldsymbol{\nu}_t) \tag{6}$$

$$\mathbf{K} = \mathbf{P}^p\mathbf{H}^T(\mathbf{H}\mathbf{P}^p\mathbf{H}^T + \mathbf{R})^{-1} \tag{7}$$

In equations (6) and (7) p denotes prediction, and u update. $\boldsymbol{\nu}_t$ is the correction variance that has to added to the observations in order to the correct the Kalman gain matrix [5]. Further $\mathbf{K}$ is the Kalman gain matrix, $\mathbf{P}^p$ is the predicted error covariance matrix of the state vector and $\mathbf{R}$ is the error covariance of the observation vector. Finally, $\mathbf{H}$ is the linear observation operator derived from assuming a linear relation in (3).

3.1. SIRF filter

The particle filter is also based on Monte Carlo sampling, the idea is to draw ensemble members or particles from the prior pdf and subsequently propagate the particles with the equation (4) similarly to the EnKF. The particles represent the prior model distribution.

The posteriori distribution is made up from particles that are assigned weights from a likelihood function that measures the closeness of the particles to the measurements. The particles are resampled to avoid numerical problems, cloning particles with large weight while particles with small weights are eliminated.

In the first experiment (4.1) the standard SIRF particle filter that will be used [25, 12]. In experiment (4.2) the concept of the importance resampling will be extended to include the localization of the importance weights. The for both the standard SIRF and the localization SIRF, it is assumed that N independent and identically distributed random samples are used to approximate posterior distribution. The empirical representation of this distribution can be written as

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})\mathbf{d}\mathbf{x}_{0:t} = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})\mathbf{d}\mathbf{x}_{0:t}, \tag{8}$$

where $\delta_{\mathbf{x}_{0:t}}$ is the delta-dirac mass located in $\mathbf{x}_{0:t}$. An alternative to the equally weighted sample is a weighted sample, represented as

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})\mathbf{d}\mathbf{x}_{0:t} = \sum_{i=1}^{N}\omega_t^i\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})\mathbf{d}\mathbf{x}_{0:t}, \tag{9}$$

where $\sum_{i=1}^{N}\omega_t^i = 1$. Propagating the particles and including the information from the upcoming observation influences these weights, and they can be written as

$$\omega_t^i = \omega_{t-1}^i\frac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t^i|\mathbf{x}_{0:t-1}^i, \mathbf{y}_{1:t})}, \tag{10}$$

where ω_{t-1} is the original importance weights, $p(\mathbf{y}_t|\mathbf{x}_t)$ is the local likelihood linking the observations and particles, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the state transition distribution and $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})$ is the proposal distribution. From a theoretical point of view the foremost property of the proposal distribution is to minimize the variance of the importance weights conditional on $\mathbf{x}_{0:t-1}$ and $\mathbf{y}_{1:t}$ [8].

Finding an efficient proposal distribution can be very difficult and most of the time the proposal distribution is identical to the dynamics of the system, $q(\mathbf{x}_t|\cdot) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$. This was used in e.g. [25, 12] and is widely used by practitioners.

For best use of the above definition the concept of weights should be elaborated. When the N particles are drawn from (4) at time $t-1$ the particles are assumed to be equally probable and therefore have the weight $\omega_{t-1}^i = \frac{1}{N}$. Equation (9) can then be rewritten as

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})\mathrm{d}\mathbf{x}_{0:t} = \sum_{i=1}^{N}\omega_t^i\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})\mathbf{d}\mathbf{x}_{0:t}, \tag{11}$$

where ω_t^i is given by (10). This is the SIS filter and unfortunately does the filter rarely work in practice as all weights

except one tends to zero as t increases. This degeneracy can be avoided by resampling the particles, replacing the weighted sample with

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})\mathrm{d}\mathbf{x}_{0:t} = \sum_{i=1}^{N} \frac{n_t^i}{N} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})\mathbf{d}\mathbf{x}_{0:t}, \tag{12}$$

where n_t^i is the integer number of clones of particle $\mathbf{x}_{0:t}^i$. The number of clones satisfies $\sum_{i=1}^N n_t^i = N$ and $E[n_t^i] = \omega_t^i N$.

In the this paper the local likelihood function has been chosen to be the multivariate Gaussian density that measures the L_2 norm between the observations and particles

$$p(y_t|x_t^i) \propto \exp\left(-\frac{1}{2}(y_t - \mathbf{H}x_t^i)^T\mathbf{R}^{-1}(y_t - \mathbf{H}x_t^i)\right) \tag{13}$$

This will also be the form of ω_t^i as the previous weight after resampling is given by $\omega_{t-1}^i = N^{-1}$ and the system dynamics is used to propagate the particles. Simple calculations the give that

$$\omega_t^i = \frac{\exp(-\frac{1}{2}(y_t - \mathbf{H}x_t^i)^T\mathbf{R}^{-1}(y_t - \mathbf{H}x_t^i))}{\sum_{k=1}^N \exp(-\frac{1}{2}(y_t - \mathbf{H}x_t^k)^T\mathbf{R}^{-1}(y_t - \mathbf{H}x_t^k))}. \tag{14}$$

After each resampling the weights have be neutralized to $\omega_t^i = N^{-1}$. This factor will be canceled with each new generation of ω_{t+1}^i. The sampling and resampling step now determines which particles survive and which particles are discarded and the particles that are discarded are resampled from the surviving particles. It should be noted that every moment having finite expectation can be approximated by integrating the using the empirical representation accordingly. For a more stringent deduction of the theory see [7].

3.2. Localization

Localization means that we only will consider particles in an immediate neighborhood in the analysis. Common practice with the EnKF is to multiply the Kalman gain with a Gaussian kernels such that the spurious long range correlations can be suppressed. The design of the localization depends on the model domain and the density of the observations in the immediate neighborhood.

With the particle filter setting we cannot use the same approach as with the EnKF, however, we can use a hybrid filter setting for the localization of the analysis step. The idea is closely related to the Auxiliary Particle Filter (APF) [21], however, here we use the idea of a proposal distribution to make a localization scheme. We will assume that we can divided the state space χ into J disjoint regions $\chi = \cup_{j=1}^J \chi^j$, where j is the index for the local region.

Introducing an arbitrary test function $\varphi(x_t) : \chi \to \mathcal{R}$ would make is possible to compute expectations of this function with respect to the posterior density

$$E_t[\varphi(\mathbf{x}_{0:t})] = \int \varphi(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})\mathbf{d}\mathbf{x}_{0:t}. \tag{15}$$

Often is the interest restricted computing expectations with respect to the filter density, i.e. computing

$$E_t[\varphi(\mathbf{x}_t)] = \int \varphi(\mathbf{x}_t)p(\mathbf{x}_t \mid \mathbf{y}_{1:t})\mathbf{d}\mathbf{x}_t. \tag{16}$$

Expanding this expression gives

$$E_t[\varphi(\mathbf{x}_t)] = \int \varphi(\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t})\mathbf{d}\mathbf{x}_t \tag{17}$$

$$= \frac{\int \varphi(\mathbf{x}_t)p(\mathbf{y}_t, \mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\mathbf{d}\mathbf{x}_t\mathbf{d}\mathbf{x}_{t-1}}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \tag{18}$$

$$= \frac{\int \varphi(\mathbf{x}_t)p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\mathbf{d}\mathbf{x}_t\mathbf{d}\mathbf{x}_{t-1}}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \tag{19}$$

Here $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum_{i=1}^N \omega_t^i \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t)$, or differently written $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum_{i=1}^N p(\mathbf{x}_{t-1}|i\mathbf{y}_{1:t-1})p(i|\mathbf{y}_{1:t-1})$. We are also interest in introducing the regions in our computations. Similar arguments suggests

$$p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) = \sum_{j=1}^{J}\sum_{i=1}^{N} p(\mathbf{x}_{t-1}, i, j \mid \mathbf{y}_{1:t-1}) \tag{20}$$

$$= \sum_{j=1}^{J}\sum_{i=1}^{N} p(\mathbf{x}_{t-1}|i, \mathbf{y}_{1:t-1})p(i|j)p(j|\mathbf{y}_{1:t-1}) \tag{21}$$

where $p(\mathbf{x}_{t-1}|i, \mathbf{y}_{1:t-1}) = \delta_{\mathbf{x}_{t-1}^{(i)}}(\mathbf{x}_t)$, $p(i|j) = \frac{1}{N_j}\mathbf{1}_{\{x_{t-1}^i \in \chi^j\}}$, $p(j|\mathbf{y}_{1:t-1}) = \frac{N_j}{N}$ and $N_j = \sum_{i=1}^N \mathbf{1}_{\{x_{t-1}^i \in \chi^j\}}$. Augmenting the space with these variables makes it possible to using importance sampling on regions rather than on particles. Including the particle index and regions in the expectation, and introducing an importance sampler for j gives

$$E_t[\varphi(\mathbf{x}_t)] = C\int \varphi(\mathbf{x}_t)p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})\cdot \frac{p(\mathbf{x}_{t-1}|i)p(i|j)p(j|\mathbf{y}_{1:t-1})}{q(j)}q(j)\mathbf{d}\mathbf{x}_t\mathbf{d}\mathbf{x}_{t-1}\mathbf{d}i\mathbf{d}j \tag{22}$$

where $C = 1/p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$. It would be trivial to introduce importance samplers to the other variables as well, but we leave that out as the purpose is to study localization. A Monte Carlo approximation of the expectation would then be given by

$$\widehat{E_t[\varphi(\mathbf{x}_t)]} = \frac{1}{N}\sum_{i=1}^{N} C\varphi(\mathbf{x}_t^i)\frac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(j|\mathbf{y}_{1:t-1})}{q(j)} \tag{23}$$

$$= \sum_{i=1}^{N} \omega_t^i \varphi(\mathbf{x}_t^i) \tag{24}$$

In this setting we have introduced the proposal distribution $q(j)$ chosen as,

$$q(j) = p(j|\mathbf{y}_{1:t}) \tag{25}$$

$$= \frac{p(\mathbf{y}_t|j, \mathbf{y}_{1:t-1})p(j|\mathbf{y}_{1:t})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \tag{26}$$

$$= Cp(\mathbf{y}_t|j, \mathbf{y}_{1:t-1})p(j|\mathbf{y}_{1:t-1}) \tag{27}$$

It can easily be seen that most of these terms cancel out, when inserted into (23). A problem is that $p(\mathbf{y}_t|j, \mathbf{y}_{1:t-1})$ is usually unknown. An approximation can be obtained from

$$p(\mathbf{y}_t|j, \mathbf{y}_{1:t}) = \int p(\mathbf{y}_t, \mathbf{x}_t, i|j, \mathbf{y}_{1:t-1})\mathbf{d}\mathbf{x}_t\mathbf{d}i \tag{28}$$

$$= \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|i)p(i|j, \mathbf{y}_{1:t-1})\mathbf{d}\mathbf{x}_t\mathbf{d}i \tag{29}$$

$$\approx \frac{1}{L}\sum_{l=1}^{L} p(\mathbf{y}_t|\xi^l) \tag{30}$$

where ξ is drawn from $p(\mathbf{x}_t|j, \mathbf{y}_{1:t-1})$. It is possible to use $L = 1$, and we denote the this approximation

$$\hat{p}(\mathbf{y}_t|j, \mathbf{y}_{1:t}) = p(\mathbf{y}_t|\xi) = p(\mathbf{y}_t|\mathcal{X}^j). \tag{31}$$

The resulting importance sampler can then be written as

$$\begin{aligned} q(j) &= \tilde{C}\hat{p}(\mathbf{y}_t|j, \mathbf{y}_{1:t})p(j|\mathbf{y}_{1:t-1}) \\ &= \tilde{C}p(\mathbf{y}_t|\mathcal{X}^j)p(j|\mathbf{y}_{1:t-1}) \end{aligned} \tag{32}$$

The 1^{st} stage weights are proportional to $\hat{p}(y_t|j, y_{1:t-1})$, cf. eq. (22) and are given by

$$\omega_{t-1}^{1st}(k) = \frac{\mathbf{1}_{\{x_{t-1}^k \in \mathcal{X}^j\}}p(\mathbf{y}_t|\mathcal{X}^j)}{\sum_{k=1}^{N}\mathbf{1}_{\{x_{t-1}^k \in \mathcal{X}^j\}}p(\mathbf{y}_t|\mathcal{X}^j)} \tag{33}$$

The particles are resampled according to these 1^{st} stage weights in order to decrease to variance of the final estimate.

The 2^{nd} stage weights can be found by studying (23). Propagating the particles according to the dynamics and using that the expectation of $\phi(x_t) = 1$ equals one by definition gives

$$\begin{aligned} &E_t[\varphi(\mathbf{x}_t)] = C\int \varphi(\mathbf{x}_t)p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})\cdot \\ &\frac{p(\mathbf{x}_{t-1}|i)p(i|j)p(j|\mathbf{y}_{1:t-1})}{\tilde{C}p(y_t|\mathcal{X}^j)p(j|\mathbf{y}_{1:t-1})}q(j)\mathbf{d}\mathbf{x}_t\mathbf{d}\mathbf{x}_{t-1}\mathbf{d}i\mathbf{d}j \end{aligned} \tag{34}$$

$$\begin{aligned} &= \frac{C}{\tilde{C}}\int \varphi(\mathbf{x}_t)p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})\cdot \\ &\frac{p(\mathbf{x}_{t-1}|i)p(i|j)}{p(y_t|\mathcal{X}^j)}q(j)\mathbf{d}\mathbf{x}_t\mathbf{d}\mathbf{x}_{t-1}\mathbf{d}\mathbf{i}\mathbf{d}\mathbf{j} \end{aligned} \tag{35}$$

$$\approx \frac{1}{N}\frac{C}{\tilde{C}}\sum_{k=1}^{N}\varphi(\mathbf{x}_t^k)\frac{p(\mathbf{y}_t|\mathbf{x}_t^k)}{p(\mathbf{y}_t|\mathcal{X}^{j(k)})} \tag{36}$$

Here, we use that $E_t[1] = 1$ to find the 2^{nd} stage weights. These are given by

$$\omega_t^{2nd}(k) = \frac{\frac{p(\mathbf{y}_t|\mathbf{x}_t^k)}{p(\mathbf{y}_t|\mathcal{X}^{j(k)})}}{\sum_{k=1}^{N}\frac{p(\mathbf{y}_t|\mathbf{x}_t^k)}{p(\mathbf{y}_t|\mathcal{X}^{j(k)})}}. \tag{37}$$

This approximation can be interpreted as two filter setting with a local filter $p(\mathbf{x}_{t-1}|j, \mathbf{y}_{1:t-1})$ only active in the local parts of the state space and a global filter $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ that is only active in the global state space but uses the local analysis as a proposal distribution.

procedure SILRF
  for $j = 1, \cdots, J$, regions
  sample index $i^j \sim p(i|j)$, and the corresponding
  particles $\zeta^j \sim p(x_{t-1}|i^j, y_{1:t-1})$
  Propagate $\xi^j \sim p(x_t|\zeta^j, y_{1:t-1})$
  Approximate $\hat{p}(y_t|\mathcal{X}^j) = p(y_t|\xi^j)$
Given that $\hat{p}(y_t|\mathcal{X}^j)$ **is known**
  for $n = 1, \cdots, N$, particles
  Resample the filter particles at $t-1$ based on
  the 1^{st} stage weights
  Propagate these to get x_t^k
  Compute the 2^{nd} stage weights and resample
  from these to get the filter sample at time t.
end procedure

Algoritme 1: The generic SILRF algorithm

In **Algorithm** 1, the Sequential Importance Localization Resampling Filter (SILRF) is described in pseudo code

After we have made the deconstruction of the domain into adequate local regions we are now able to perform the localization step. The localization of the importance sampling is divided into two steps. **Step 1**: we calculate the importance weights from the particles using eq. (33) and (37). With the found local weights we will now resample each particle in this local region. After the resampling step each particle is has the $\frac{1}{N}$ probability of being selected. In other words we discard the particles in this local region that have little importance to the observations and resample from the particles that survived the selection process. This is done for all the local regions.

Step 2: Secondly we calculate the global importance weights from the entire domain without the deconstructions of the domain by sampling from the distribution eq. (37). Using the observation twice is not a problem with the particle filter. Since we only adjust the probability that we have assigned to them in the importance weighting process. We are now using the already predetermined particles as a proposal distribution to the global importance weights which then will correct for the use of the observation twice.

The interpretation is that we have already determined which particles are adequate in the local regions and discarded the improper particles. This means that the only particles that are left are the particles that at least fit the observations in the local regions. In the global domain we now have to select the particles from the proposal distribution that will fit the entire domain. However, we cannot be sure that these all particles will have a meaningful representation in the global domain. Therefore we will have to weight the particles again with respect to all observations in the global domain. This step will select the particles from the proposal distribution that will have the best representation in the global domain. This step will also reduce the noise in the system since it can be seen as smoothing step between the local regions.

The proposed particle filter SILRF is much cheaper to use in a high dimensional space than the auxiliary particle filter. In the APF filter we would have to calculate $p(y_t|\cdot)$, for N particles

whereas in the proposed SILRF we only have to calculate the $p(y_t|\chi^j)$ for J candidates to get the $1st$ stages weights. This clearly much more computational efficient.

3.3. Configuration of the filters

The SIRF filter is used in this paper as it is the simplest of the PFs. There are many other PF formulations in the non-linear filtering community. However, the SIRF is fairly simple and it is widely used. Comparing different PFs is a topic of its own, which will not be addressed in this contribution, where focus is on the effectiveness of using localization of the importance weights. For other flavors of PFs a good starting point is [7]. The EnKF is only used in this paper as reference and a benchmark for the performance of the SIRF.

All filters are configured in the same way for an optimal comparison. The configuration is described in the following. The model noise is Brownian motion with transition variance $B_t - B_{t-dt} \sim N(0, dt)$, where dt is the integration step of the Runge-Kutta scheme and has the value of $dt = 0.01$. The diffusive parameter $\sigma = 4.0$. The observation error covariance matrix is $\mathbf{R} = 10.0 \cdot \mathbf{I}$ and a realization of observation error was added to the observations derived from a single realization of the sL95 to construct the observation time series. The value of the observation error covariance matrix was chosen such that the observation error was within 0.25 of the range of the model dynamics. The measurement operator $\mathbf{H} = \mathbf{I}$, i.e. all nodes are observed and only the time interval between observations is varied. Effects on limiting the observability of the system is beyond the scope of this paper.

The prediction performance is measured by the mean squared error (mse) of the n-step prediction. The measure of performance is achieve by looking at predictability of the system. The predictability is investigated by varying the update frequency, and the mse is calculated from the true solution against the assimilated solution.

4. Experiments and results

First the sL95 is initialized and run for a period of 64,000 time steps. The observations consist of measurements drawn from the sL95 for 8 nodes as one realization from the stochastic process. The initial values were also stored for use in the assimilation experiments. The filtrations are started from the same initial state to stabilize the filters. The assimilation experiment is set up in the following way. The EnKF operates with 250 ensemble members and the solution has converged for this number of ensembles. The prediction performance did not improve significantly when the number of ensembles was increased to 500 or 1000 members. However, the SIRF solution did not converge with 250 particles, it had to have a particle size of 1,500 members to converge on the solution. This behavior is typical of the SIRF since the statistics of the solution must be calculated based on the filter pdf. The filter pdf is non Gaussian in most cases, and in order to get a good representation of the complete filter pdf, more particles are needed than with the EnKF, which only uses linear statistics, i.e. the mean and the (co)variance of the ensembles members.

The experiments are conducted as follows, a) first the sL95 is examined for a small setup to investigate the performance of the filters. From the results the experiment b) investigates the impact of localization as the dimension of the state space increases.

4.1. Experiment 1

In Figure (2) time series of the true solution, observations at varying time interval and corresponding filtering solutions using the EnKF and SIRF are shown. Time series are depicted for update intervals of $\{10, 20, 30, 40\}$.

All following analysis will be limited to prediction step less than of 50. The justification for this can be found by analyzing Figure (2). The reconstruction deteriorates as the update interval goes through the updates intervals $\{10, 20, 30, 40\}$. For update intervals of 10 and 20, the reconstruction is very decent and the filters are able to reconstruct the dynamics of the of the observations. However, the reconstruction of the filters is worse for maintaining the dynamics of the observed time series for update intervals of 30 and 40. Its clear that the filters looses control over the reconstruction and if the update interval is increased beyond 50 the predictability is lost.

When the system is densely observed, the assimilation problem is nearly linear and Gaussian statistics applies well, [24] and [27]. In other words the propagated ensemble members are not spread too much in the state space and therefore the Gaussian model is still a good approximation of the filter pdf. In [16] an experiment with partial limited observations of the state space was conducted using the Lorenz attractor. It showed that the SIRF outperforms the EnKF because of the the fully nonlinear update step. In this paper, systems are investigated focusing on demonstrating the effect of localization of the importance sampling to restore prediction performance for the SIRF for medium sized state spaces. A natural next step is to exploit the scalability of the SILRF and the skill of the SIRF for spatially sparse observed system to obtain a fully non linear and computationally feasible filter.

Detailed analysis of Figure (2) shows subtle differences between the SIRF and the EnKF in the way of handling the statistics of the assimilations. There is a nice discussion in [25] on similar differences when updating the states for the Kortewegg-DeVries equation for the SIRF and the EnKF. However, in the mse sense the two filters performs equally well for the sL95 system as can be seen from figures (3) and (4).

We have chosen to represent the mse of the n-step prediction as box plots. The mse is calculated for each node in the system over all the particles used in the simulation. In the Figures (3) and (4) the mse is depicted in a box plot setting. The red line in the boxes indicates the median and the lower and upper box edges indicates lower and upper quartiles respectively. The whisker indicate lowest and highest non outliers and the pluses indicates outliers.

4.2. Experiment 2

In this experiment the idea is to investigate the impact of using localization to reduce the importance sampling dimension

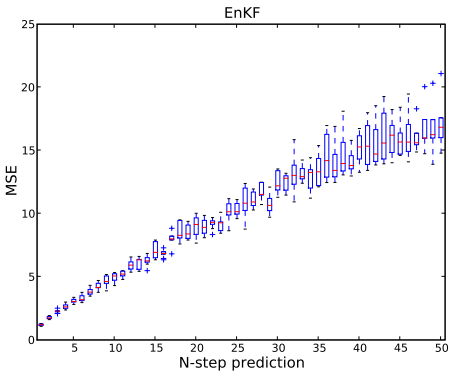


Figure 3: The mean squared error of EnKF n-step prediction for $n \in \{1, 50\}$. The box plot shows how the mse grows throughout the nodes as a function of the n-step prediction

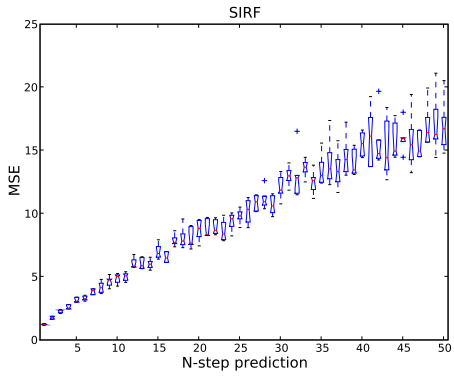


Figure 4: The mean squared error of SIRF n-step prediction for $n \in \{1, 50\}$. The box plot shows how the mse grows throughout the nodes as a function of the n-step prediction

for medium sized state spaces. This is done by dividing the sL95 ring into smaller domains. In the experiment each local region will only have 8 nodes within the local region. The choice for number of local region nodes is based on experiment (4.1), where it was shown that the SIRF has a performance equal to EnKF for 8 nodes. The conjecture is that the performance (in the mse sense) of experiment (4.1) is maintained for increasing state space size when using localized importance sampling. By applying the importance sampling localization to the smaller subspaces, the SIRF filters should be able to maintain the performance of experiment (4.1) if the proposed methodology is successful in reducing the effective dimension as assumed.

4.3. How the SILRF works in practice

Before we start on the main experiments we would like to give a more intuitive feel to the SILRF. We will demonstrate how the SILRF will be able to give a better estimate of the empirical prediction distribution then the SIRF.

In this experiment will use the sL95 with 48 nodes and 6 local regions each containing 8 nodes. The setup is depicted in Figure 5. We will use a update step of 35 time steps between each observation. In this experiment we have used the SILRF as the main filter, we have updated the model up to time step 385. Now at the next update step at $t = 420$ we will use both SIRF and the SILRF and look at the prediction distribution and the local proposal distribution and the filter distribution for node 45. The node 45 is in the center of the region 6 in the Figure 5. The distributions in the following figures are shown as red kernel density estimates over the particles. The observations is shown as green Gaussian bell $\sim N(y^{45}_{420}, R)$. The particles are shown as blue | on the x-axis; note that after a resampling step many of the particles may have the same value and therefore many of the blue markers | are overlapping. The vertical black line indicates the state estimation calculated from the importance weights. In Figure **6a** the prediction distribution is shown for both filters; the prediction distribution is the same for both filters. We see that the prediction distribution and the observations does not overlap each other very well and this will eventually lead to a very poor filter estimation since many of the particles will be discarded and only few will be resampled. However, for the SILRF we have the intermediate local sampling step. The proposal distribution for node 45 is shown in Figure **6b**, here we can see that we successfully have shifted the prediction distribution towards the observation and therefore will the proposal distribution have better support for final update step with the SILRF. The vertical black line indicates the state estimation if we have only chosen to use the local weights importance weights. Looking at Figure **6c** the filter distribution for the SIRF together with the observation did not match very well and the state estimation is not close to the observation. The filter distribution from the SILRF in Figure **6d** is the same as the one we calculated in the local region. However, the global importance weights have be modified according to eq. (22) and therefore we get a different state estimate. The global state estimation is slightly different from the the state estimate in the local region and should be a better estimate since the global importance weights is calculated from the entire system with the the locals proposal distribution.

4.4. Experiments with 4 different setups

In the experiment, the sL95 is expanded to $n = (24, 46, 72, 96)$ and hence $m = (3, 6, 9, 12)$ local regions. Be reminded that m could also be any integer number for which $m \geq 1$, e.g. if $m = 1$ then the local region is the global domain. In the illustration of Figure (5) the sL95 has 48 nodes and 6 local regions marked with an number from 1 to 6.

The results from the experiments with the SILRF are shown in Figure (7). From the top left to the bottom left corner results from the sL95 with $n = (24, 48)$ are shown and in the right column from top to bottom the results with $n = (72, 96)$ are shown.

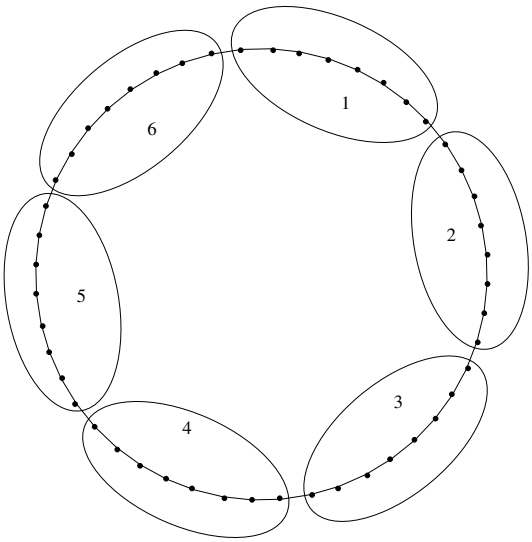


Figure 5: The sL95 ring divided into the six local regions containing 8 nodes. Each local regions is marked with a number from 1 to 6

For comparison the n-step prediction mse is also shown for the SIRF without the localization. The results without the localizations are shown are as squared boxes. The notched boxes are results from the SILRF.

From Figure (7) it can be seen that the SIRF as expected does a poor job without the localizations as the dimension of the sL95 increases. Starting with the Figure 7a for $n = 24$ there is still some prediction performance left in the SIRF for the smaller system and therefore the n-step mse does not grow as fast. The effects of the localization is however evident with a 50% reduction of mse for intermediate prediction horizon. Further, the whiskers of for the SILRF are smaller than those of the SIRF and the error growth is linear in the system. The effects of the localization becomes even more evident when the sL95 is increases to 48 and beyond which can be seen in the Figures 7b, 7c, 7d. The localization of the importance sampling is clearly measurable in the system as the predictability skill of the SIRF is non existent after a few steps. The n-step mse grows fast and quickly flattens out which indicates that the SIRF has lost control over the reconstruction and that model and observations are out synchronization. In all figures the error growth of the SILRF has a much more linear tendency compared to the SIRF and does not grow with the dimension. It should also be noted that the SILRF reduces the noise in the system for every prediction horizon. This can be seen from the width of the whiskers for respectively the SIRF and the SILRF. The width of whiskers indicates the total variability of the mse in the system. The total noise in the system is reduced even though the importance sampling decomposed the domain into local regions. It could be feared that the use of local regions would introduce strong gradients in the transition from local region to the next local region. However, the narrow whiskers demonstrates a homogeneous set of mse values throughout the domain. Thus, this is clearly not the case and the SILRF performs very well compared to the SIRF.

The SIRF reaches stationary variance after 15 steps whereas the SILRF has a linear error growth and can handle prediction horizons much larger then ≥ 50. This is clearly an indication of the robustness of the localization works.

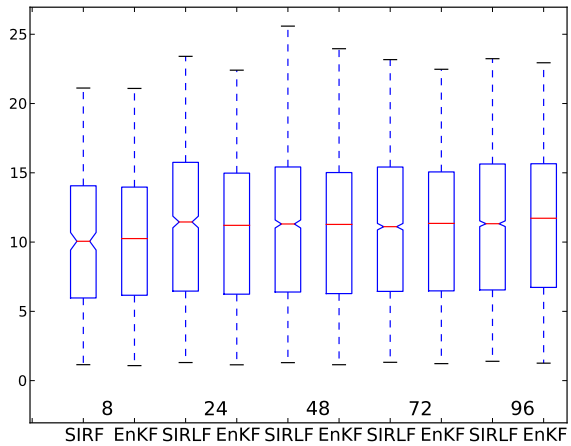


Figure 8: The boxplot shows the MSE values for all 50 values from the respective figures 7 as one box. The MSE values of the SILRF and EnKF is shown together for the respective sL95 simulations. The two first boxes are the reference run of the SIRF and the EnKF for the small sL95 with 8 nodes. The label number on the x axis indicates the dimension of sL95 simulation, and the name indicates which filters was used. The particle filters are shown as notched boxes and the EnKF is shown as rectangle boxes.

5. Conclusions

It has been shown that the Sequential Importance Localization Resampling Filter (SILRF) improves the predictability in the system for models with state space sizes of orders up to 10^2. The Sequential Importance Resampling Filter (SIRF) was compared to the EnKF on a small L95 system with 8 nodes. On this system the EnKF and the SIRF had the same prediction performance in the mse sense. With a starting point in this result experiments with 4 new L95 systems with size $n = (24, 48, 72, 96)$ were carried out. Each of these domains was divided into smaller domains in the localized importance sampling experiments. Each of the local regions had 8 nodes. On these L95 system the SILRF proved to have an almost linear error growth for all state space sizes. The SILRF was compared to the SIRF which showed an exponential error growth as a function of the n-step prediction horizon. This exponential

error growth became more clear as the system dimension increased and gave evidence to the curse of dimensionality as the number of particles was maintained. The SILRF was also able to decrease the mse variability among the nodes of the system at every prediction step contrary the SIRF. This demonstrates that the potential discontinuities introduced by reconstructing the state from the local region representations are not problematic for applying SILRF to the L95 system.

The SILRF method proved itself as a robust filter under the right conditions. The simplicity of the SILRF makes it easy to implement in models with intermediate state space sizes for which non-linear modeling is a key issue. At the moment there is no reason to believe that this approach should not work on larger system then the ones suggested in this paper.

Acknowledgements

This work has been financially supported by the Danish Research School ITMAN

[1] Alspach, D. and Sorenson, H.: Nonlinear Bayesian estimation using Gaussian sum approximations, IEEE Transactions on Automatic Control, 17, 439–448, 1972.

[2] Bengtsson, T., Snyder, C., and Nychka, D.: Toward a nonlinear ensemble filter for high-dimensional systems, Journal of Geophysical Research, 108, STS2–1–10, 2003.

[3] Bengtsson, T., Bickel, P., and Li, B.: Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems, IMS Collections - Probability and Statistics in Honor of David A. Freedman, 2, 316–334, doi:10.1214/193940307000000518, 2008.

[4] Berliner, L. M. and Wikle, C.: Approximate importance sampling Monte Carlo for data assimilation, Phys. D, 230, 37–49, 2007.

[5] Burges, G., van Leeuwen, P. J., and Evensen, G.: Analysis Scheme in the Ensemble Kalman Filter, American Meteorological Society, 126, 1719–1724, 1998.

[6] Burrage, K. and Burrage, P. M.: High strong order methods for non-commutative stochastic ordinary differential equation systems and the Magnus formula, Phys, D, 133, 34–48, 1999.

[7] Doucet, A., de Freitas, N., and Gordon, N.: Sequential Monte Carlo Methods in Practice, Springer Verlag, first edn., 2001.

[8] Doucet, A., Gordon, N. J., and Krishnamurthy, V.: Particle Filters for State Estimation of Jump Markov Linear Systems, 2001.

[9] Evensen, G.: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, Journal of Geophysical Research, 99, 10 143–10 162, 1994.

[10] Evensen, G.: The Ensemble Kalman Filter: theoretical formulation and practical implementation, Ocean Dynamics, 53, 343–367, 2003.

[11] Gordon, N. J.: Non-linear/Non-Gaussian Filtering and The Bootstrap Filter, Non-Linear Filters, IEE Colloquium on, pp. 4/1–4/6, 1994.

[12] Gordon, N. J., Salmond, D. J., and Smith, A. F. M.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation, Radar and Signal Processing, IEE Proceedings F, 140, 1993.

[13] Hamill, T. M., Jeffrey, Whitaker, S., and Snyder, C.: Distance-Dependent Filtering of Background Error Covariance Estimates in an Ensemble Kalman Filter, 2776 MONTHLY WEATHER REVIEW VOLUME 129 Monthly Weather Review, 129, 2776–2790, 2001.

[14] Houtekamer, P. L., Mitchell, and Herschel, L.: Data Assimilation Using an Ensemble Kalman Filter Technique, Monthly Weather Review, 126, 796–811, 1998.

[15] Kalman, R.: A new Approach to Linear Filtering and Prediction Problems, Journal of Basic Engineering, 82 D, 35–45, 1960.

[16] Kivman, G. A.: Sequential parameter estimation for stochastic systems, Nonlinear Processes in Geophysics, 10, 253–259, 2002.

[17] Lorenz, E. N.: Predictability: A problem partly solved, in: Seminar on Predictability at ECMWF, vol. 1, pp. 1–18, 1995.

[18] Lorenz, E. N. and Emanuel, K. A.: Optimal Sites for Supplementary Weather Observations: Simulation with a Small Models, Journal of the Atmospheric Sciences, 55, 399–414, 1998.

[19] Losa, S. N., Kivman, G. A., and Ryabchenko, V. A.: Weak constraint parameter estimation for a simple ocean ecosystem: what can we learn about the model and data?, Journal of Marine Systemes, 45, 1–20, 2003.

[20] Losa, S. N., Kivman, G. A., Schroter, J., and Wenzel, M.: Sequential weak constraint parameter estimation in an ecosystem model, Journal of Marine Systemes, 43, 31–49, 2003.

[21] Pitt, M. K. and Shephard, N.: Filtering via Simulation: Auxiliary Particle Filters, Journal of the American Statistical Association, 94, 590–599 and 2670 179, 1999.

[22] Rabier, F.: Overview of global data assimilation developments in numerical weather-prediction centres, Quarterly Journal of the Royal Meteorological Society, 131, 3215–3233, 2005.

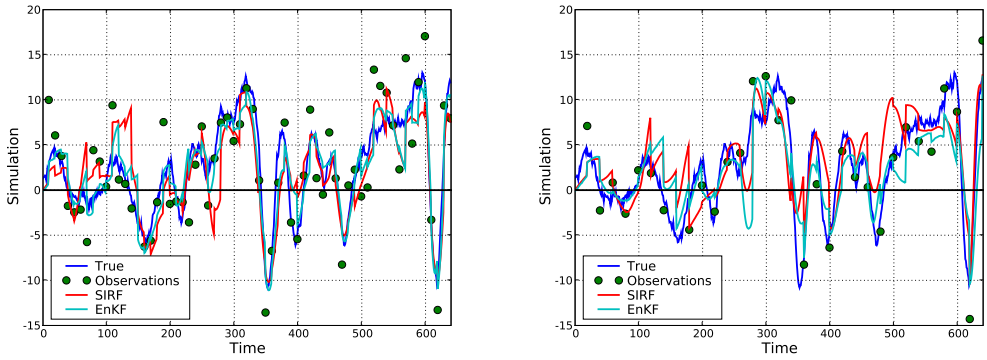
[23] Sørensen, J. V., Madsen, H., and Madsen, H.: Efficient Kalman filter techniques for the assimilation of tide gauge data in three-dimensional modeling of the North Sea and Baltic Sea system, Journal of Geophysical Research - Part C - Oceans, 109, 14, URL `http://www2.imm.dtu.dk/pubdb/p.php?3618`, 2004.

[24] Sørensen, J. V. T., Madsen, H., and Madsen, H.: Data assimilation in hydrodynamic modelling: on the treatment of non-linearity and bias, Stochastic Environmental Research and Risk Assessment, 18, 228–244, 2004.

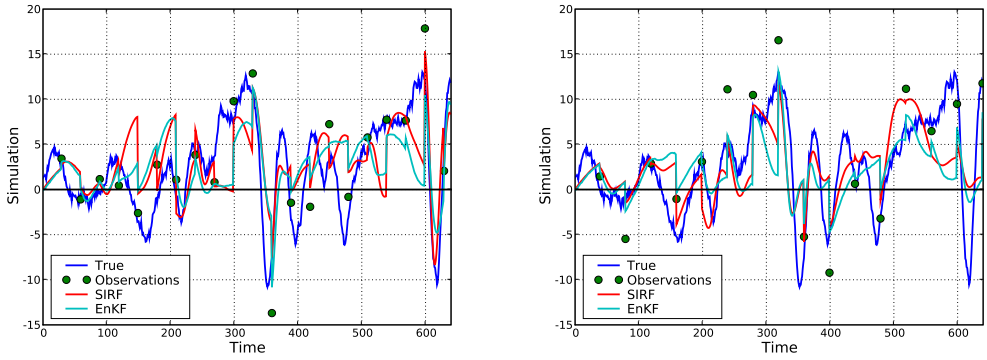
[25] van Leeuwen, P. J.: A Variance-Minimizing for Large-Scale Applications, American Meteorological Society, 131, 2071, 2003.

[26] van Leeuwen, P. J. and Evensen, G.: Data Assimilation and Inverse Methods in terms of a Probabilistic Formulation, American Meteorological Society, 124, 2898, 1996.

[27] Verlaan, M. and Heemink, A.: Nonlinearity in Data Assimilation Application: A Pratical Method for Analysis, Monthly Weather Review, 129, 1578–1589, 2001.



(a) t=10 **(b)** t=20



(c) t=30 **(d)** t=40

Figure 2: Experiment 1: the figures show the assimilation of the sL95. From the top left to the bottom right the update frequency is $\{10, 20, 30, 40\}$. The noisy measurements are denoted with a green dot

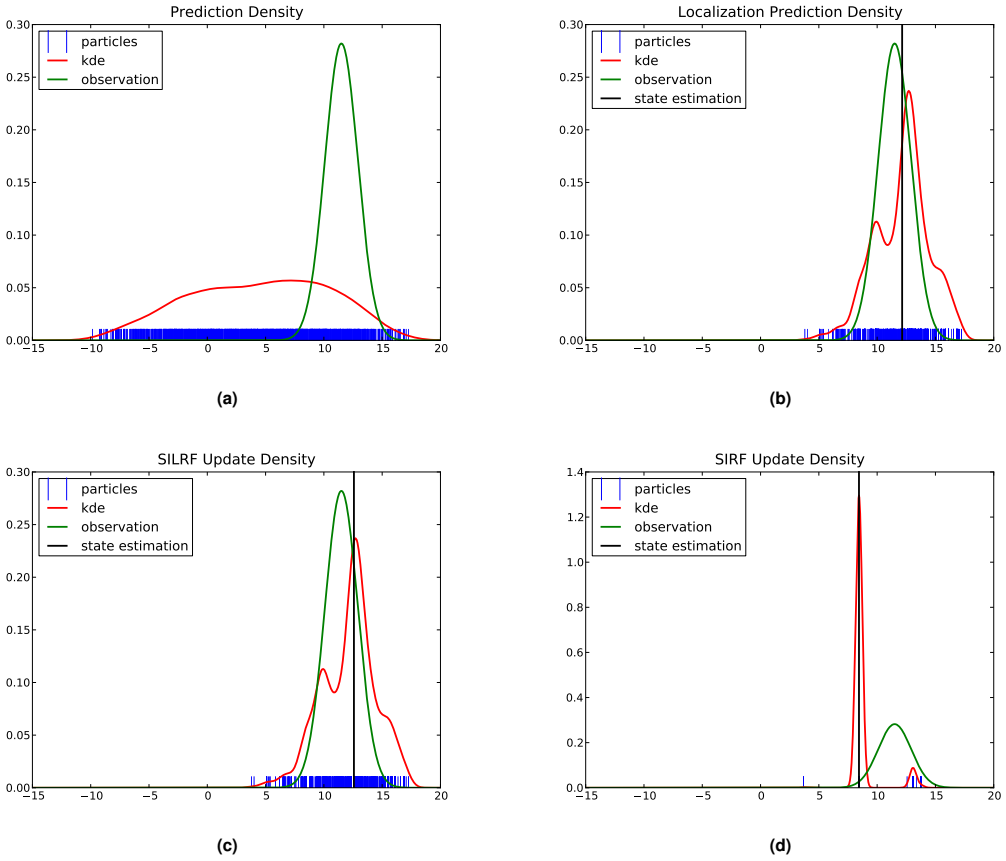


Figure 6: All the distributions are shown as kernel density estimates of the particles at different stage of the filtration. The particle distributions are shown with red. The observations is also given as a Gaussian kernel $\sim N(y_{420}^{45}, R)$. The unique particles in the estimation is shown as blue | on the x-axis. **6a** is the prediction distribution before the update. **6b** is intermediate proposal distribution from the given local region. **6c** is the filter distribution after the final update with the SIRF. **6d** is the filter distribution after the final update of the SILRF.

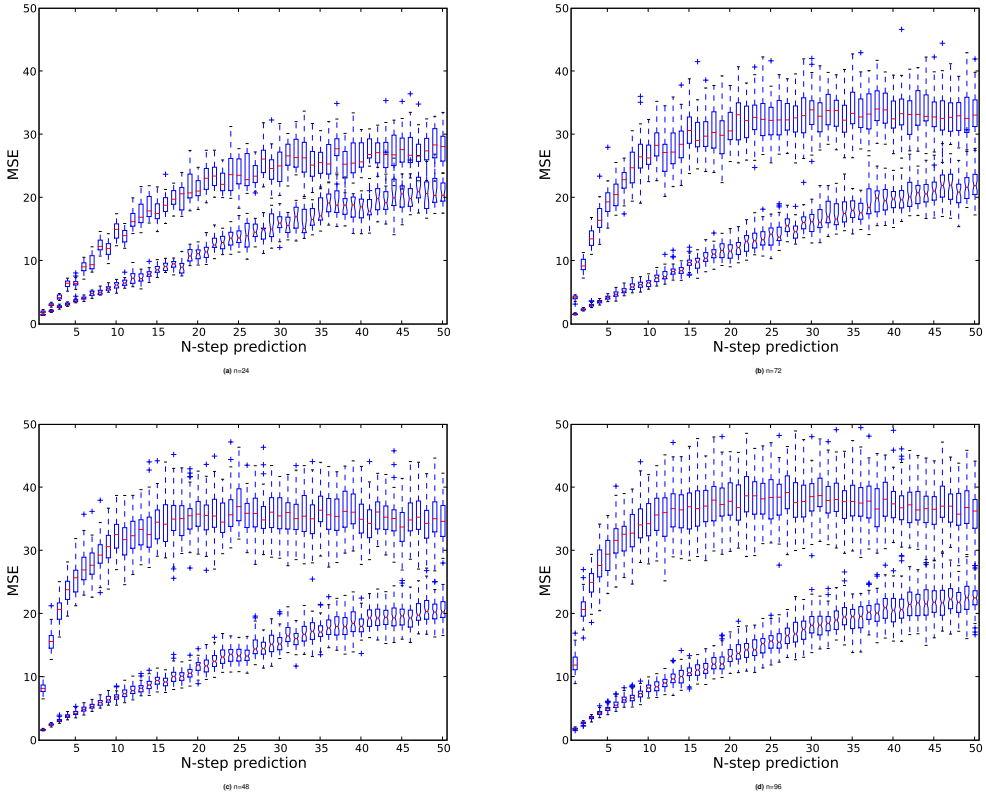


Figure 7: Results from the SILRF are shown for the sL95 with dimensions $n = (24, 48, 72, 96)$. For comparison are the n-step prediction mse shown for the SIRF without the localization. The results without the localizations are shown are as squared boxes. The notched boxes are results from the SILRF

APPENDIX B

Paper B

Submitted to Journal of Hydroinformatics

Calibration and Estimation of Biogeochemical Models of a Complex EcoSystem

Jan Frydendall[a], Madsen, H.[a], Erichsen, A.[b], Sørensen, J. V. Tornfeldt[b]

[a] *DTU Informatatics, Richard Petersens Plads,DTU,Building 321, DK-2800, Lyngby, Denmark*
[b] *DHI, Agern AllÉ 5, DK-2970, Hørsholm, Denmark*

Abstract

Traditionally model parameters are calibrated using a so-called output error method. The output error method has the disadvantages that we are not able to differentiate between model and observation errors in the estimations. This often lead to non optimal parameter values. In this paper we will present a method to estimation of parameters in a fairly simple lumped lake ecosystem model. The lumped ecosystem model is put into a stochastic differential equation framework to ensure that the parameters can be more optimal estimated through the maximum likelihood setting. The lumped lake ecosystem model is a simplified representation of the biogeochemical ECOLAB model from the DHI software suite. The measurements for the estimation are based on a measuring campaign from 2001 around the Esrum Lake in North Zealand, Denmark. In the campaign a phosphorus and inorganic nitrogen was measured on a monthly basis. In the paper we show that by embracing the stochastic differential framework we can estimate many of the parameters in the lumped model and give an estimated of the model and observation errors. We will use the CTSM semi automatic estimation toolbox for Continues Time Stochastic Modeling.

Key words: Stochastic differential equations, estimation, CTSM, eco system, lake modelling

1. Introduction

Eco system modelling is a complex and challenging task which so far has only been partly solved. The involving processes in the eco systems are often very complex and non-linear. In order to fully understand the process in

eco system high resolution numerical models are needed for diagnosing the interaction of the bio-geochemical spices and to forecast important events. In order to facilitate these models the involving parameters in the models have to be estimated. In principle all numerical models should be estimated through experimental data from observations. However, there can be several reasons why this is not possible, lack of empirical data or too sparse data which makes estimation impossible or due to scaling problems. In such situation lumped models can be considered. Lumped models are simplified models of the large numerical models and they often have a course resolution. By considering lumped models information can be estimated from observed data that is more suited the lumped models. After the lumped models are estimated the large numerical models can be calibrated from the output data from the lumped models (Madsen, 2003). The state space model in this case described through a set of coupled ordinary differential equations here denoted ODE. The ODE of the system consider in this paper will be described in Section 3.

However, when it comes to estimating/calibratration of the numerical models only output error methods are often considered. Output error methods can best be described through the standard writing of the state space equation with discrete observations

$$\frac{d\mathbf{x}_t}{dt} = f(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\theta}_t) \tag{1}$$

$$\mathbf{y}_{t_k} = h(\mathbf{u}_{t_k}, \boldsymbol{\theta}_{t_k}, \mathbf{x}_{t_k}) + \boldsymbol{\varepsilon}_{t_k}, \tag{2}$$

where ε_{t_k} is a zero mean normal distributed white noise with a given variance, $\mathbf{x}_t$ is the state equation at time t and $\mathbf{y}_{t_k}$ is the observation at time t, and the subscribed k indicates that the observation are discrete. $\boldsymbol{\theta}$ are the parameters of the model. Output error methods (Young, 1981) is the method of minimizing the squred observation error through,

$$\boldsymbol{\varepsilon}_{t_k} = \arg\min_{\boldsymbol{\theta}\in\Theta}(\mathbf{y}_{t_k} - h(\mathbf{u}_{t_k}, \boldsymbol{\theta}_{t_k}, \mathbf{x}_{t_k}))^2,$$

with respected to the argument $\boldsymbol{\theta}$. The minimization can be archived through various optimization algorithms, such as simplex search, gradient desenct methods such as Newton-Rapton, stochastic approximation and others (Madsen, 2000). The Output Error methods, however, has a tendency of giving bias results as the random effects are influencing the parameter estimates, especially if the model structure is incorrect (Kristensen et al., 2004).

However, the output error method does not take into account that there can be auto correlation in the observation data. In the state space model (1) -(2) it is assumed that the observations are spread around the solution as white noise. This is seldom the case and it is known from statistics that in many applications that the observations often are clustered on either above our below the ODE solution of state space model. When the model is estimated/calibrated to the data the solution will always be a mean values solution. This means that the ODE solution is the best fit through the observation points even if there is auto correlation in the data. This on the other will give unnecessary large prediction errors. However, if the state space model is extended to include some model error in the system equation, then the resulting differential equation would be described as a stochastic differential equation (SDE). SDEs have a natural affinity to incorporate random effects into the system as uncorrelated increments (Øksendal, 2005; Gardiner, 2004). If a SDE based state space is chosen these random effects can be spilt into measurement noise and processes noise.

Estimating model parameters through the intensive use of SDEs is one of the corner stones of (stochastic) *Grey-box* modelling (Madsen and Melgaard, 1991; Melgaard and Madsen, 1991; Bohlin and Graebe, 1995; Bohlin, 2006). The key idea of grey-box modelling is to find the simplest model that will provide and adequate description of the variations given time series. By incorporate prior physical knowledge in the model construction phase combined with information from the experimental data, the construction phase of the model is improved considerably. However, in this paper the main focus is on calibration/estimation of parameters in numerical models through the extensive use of SDEs.

In this paper a simplified eco model will be put into the stochastic differential equation setting for parameter calibration through the maximum likelihood principles. The simplified model will be validated using the more complex state-of-the art white-box model eco model: *ECOLAB* (Lessin and Raudsepp, 2006). This paper will address the importance of maximum likelihood principles by embracing the stochastic formulation of differential equations. The theory of expanding the model to a stochastic formulation will be introduced and explained together with the theory of maximum likelihood for calibration of parameters in numerical models. Results from the calibration of the parameters will be shown and discussed. Finally the paper will be concluded with a discussion.

2. Motivation

Before, the theory gets developed a small example is given. Consider a given time series of observation that describes an exponential decay. Exponential decay is often encounter in biological system, such as eco systems. Traditionally the exponential decay is only model through a ODE

$$\begin{aligned} \frac{dx}{dt} &= -ax \\ y_{t_i} &= x_{t_i} + \varepsilon_{t_i} \end{aligned} \tag{3}$$

As stated in the above it will would be more advantageous to look at the SDE representation of the exponential decay, (Ornstein-Uhlenbeck process), where the process contains an extra stochastic diffusion term.

$$\begin{aligned} dx &= -axdt + \sigma dt \\ y_{t_i} &= x_{t_i} + e_{t_i} \end{aligned} \tag{4}$$

In Figure 1a a noisy exponential decay is shown. The blue dots represents the noisy measurements of the exponential decay. The green curve is the normal ODE representation of the exponential decay and the red curve is the stochastic exponential decay. Both the green and red curve has been calibrated to the measurements through the output error and the maximum likelihood method respectively. The ODE representation is the mean value curve of the exponential decay. On the contrary the SDE solution takes into account that there are some auto correlation in the residuals. In Figure 1b the 1 step prediction with the prediction error is shown. The definition of the coloring is the same. The SDE representation has smaller prediction error than the ODE. All the uncertainties of the measurements can only be represented through the measurement error e_{t_i}. The SDE representation has a smaller prediction error, since some of the measurement error has been identified as system error. The diffusion term σ now describes the auto correlation of the measurements errors. This relationship helps reducing the prediction errors since much of the measurement error has been reduced to system errors.

3. The lumped model

The ECOLAB model is a model-template that can be selected from the DHI (`www.dhigroup.com`) software suite. Normally the ECOLAB template

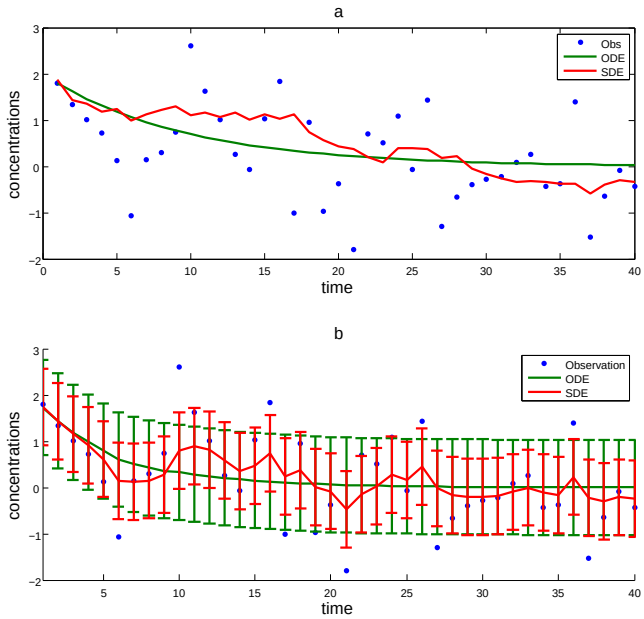


Figure 1: a) The noisy measurements with fitted curves from the ODE and SDE representation is plotted as respectively blue dots, green and red curves. b) The prediction error shown as error bars for respectively the ODE and SDE. c) The auto correlation of the residuals for the ODE and SDE.

is linked with a hydrodynamic model in order to simulate transport of the bio-geochemical species in water environments. The lumped model will be a zero spatial dimensional model with only the temporal state equation which involves interaction between the bio-geochemical species. ECOLAB is a complex model with many states and parameters. The state variables in ECOLAB include phytoplankton carbon (C), nitrogen (N), phosphorus (P), zooplankton C, detritus C, N, P, inorganic N and P, dissolved oxygen and chlorophyll-a. The interactions in ECOLAB are shown in Figure 2. Nitrogen in form of wash-out from nearby land and rivers and wet/dry depositions from the atmosphere are forcing the model. Solar radiation is also a major forcing in the model, since light and heat are catalysts for plant and algae growth. If all states and parameters are embedded in the model, then there are up to approximately a hundred parameters in the model which governs

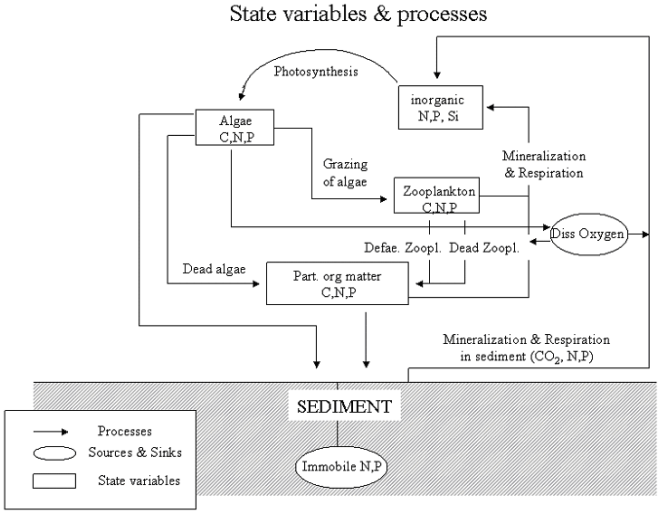


Figure 2: The interaction between the bio-geochemical species in ECOLAB.

the growth rates of the states and the coupling between the state equations. However, these parameters are often fixed in the model and the calibration of the number of model parameters is reduced to 10-20 parameters.

In the simplified model, i.e. grey-box model, only three states and 19 parameters are considered. The dynamics of the states are descriped by the following equations:

$$\frac{dC_{PN}}{dt} = PP - DEATH - SEPN \tag{5}$$

$$\frac{dC_{DN}}{dt} = DEATH - MINE - SEDN \tag{6}$$

$$\frac{dC_{IN}}{dt} = MINE + SIN - PP, \tag{7}$$

where

$$
\begin{aligned}
MINE &= K_M \theta_M^{(T-z_0)} C_{DN} \frac{C_{DN}}{C_{DN} + K_{DN}} \\
DEATH &= K_D \theta_D^{(T-z_0)} C_{PN} \frac{C_{PN}}{C_{PN} + K_{PN}} \\
PP &= \mu_{max} \theta_{PP}^{(T-z_0)} F(I) \cdot F(IN) \cdot RD \cdot FAC \\
SEDN &= K_{DN} C_{DN} \\
SEPN &= K_{PN} C_{PN} \\
SIN &= K_s (SEPN + SEDN) \theta^{(T-z_0)} \\
F(I) &= \frac{I}{I + K_I} \\
F(IN) &= \frac{C_{IN}}{C_{IN} + K_{IN}} \\
I &= I_0 \exp -(K_{I_{PN}} C_{PN} + K_{I_{DN}} C_{DN} + 0.15)
\end{aligned}
\tag{8}
$$

The simple eco system model is shown in Figure 3. The arrows represents the forcing in the eco system model i.e. nitrogen (N) and the solar radiation (PAR). The three states PN, DN, and IN are all nitrogen components. PN is the Phytoplankton Nitrogen, DN is Detritus Nitrogen, and IN is Inorganic Nitrogen. The parameters include, MINE: mineralization of detritus, DEATH: extinction of phytoplankton, PP: primary production, SEDN: sedimentation of detritus, SEPN sedimentation of phytoplankton, SIN: mineralization of sediment, F(I): solar radiation ratio, F(IN): inorganic nitrogen ratio and I: the depth averaged available photosynthetic light in the water column.

The growth of phytoplankton is controlled by nutrients, i.e. nitrogen, sedimentation, temperature and solar radiation. Bacterial degradation of organic matter in the water phase and in the sediment releases inorganic nutrients which leads to new production of phytoplankton.

3.1. Data

The biogeochemical measurements were conducted in 2001 at Esrum Lake in Denmark (56.02°, 12.40°). Esrum Lake is Denmarks second largest and water richest lake. The Lake has and area of $17,29 km^2$ and a water volume of $233 \cdot 10^6 m^3$. The Esrum Lake has characteristic shape with an even increasing depth until approximately 5 meter, thereafter there is a steep slope until

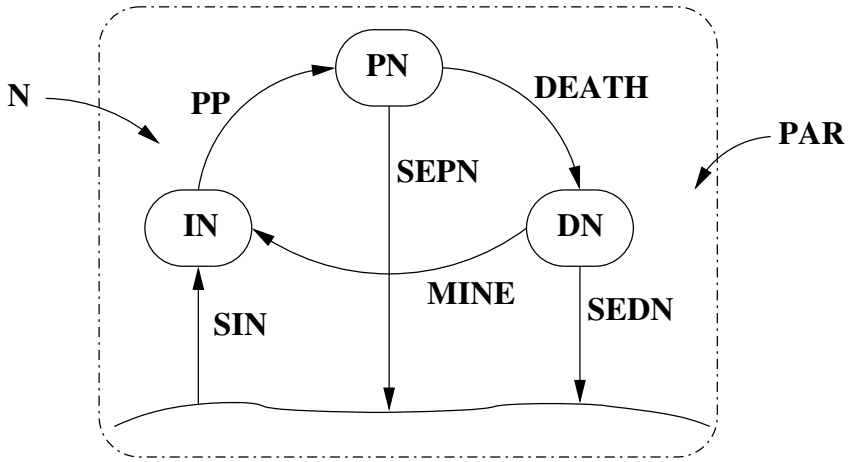


Figure 3: The interaction of the nitrogen components in the simple eco system model

approximately 15 meter, where the slope is decreasing and becomes evenly large flat bottom area. Over half of the lake area the depth is over 15 meters.

The Esrum Lake has been monitored in the years 1997-1998 and again in year 2001. The monitoring was conducted on the basis of a environment investigation into the general state of the lake after the environment protection law. The general health of the lake was described as stable even if there was an increase in phosphorus and phytoplankton biomass Frederiksborg Amt (2001). The same conclusion was derived in 2002 as descriped and compared in the Frederiksborg Amt technical report on the Esrum SøFrederiksborg Amt (2002). In 2001 the phosphorus and inorganic nitrogen biogeochemical components were measured at a monthly basis in the lake. The phosphorus biogeochemical component is measured for all 12 months of 2001. However, inorganic nitrogen is only measured from April to November in year 2001. There were also conducted other measurements at that time. However, these measurements are not related to the lumped eco lake model and therefore outside the scope of the article.

The air temperature profile from that particular site could not be obtained. Hence another air temperature profile was used that was measured within a distance of approximately 50 km from the lake at the Danish Mete-

Figure 4: A google earth image of the Esrum lake, Denmark, (56.02˚, 12.40˚)

orology Institute, Kastrup Airport measuring station. The air temperature profile is a monthly average and therefore it is judged that the offset in the distance between the two measuring stations could be neglected. The solar radiation profile was taken from a site near Sorø, Denmark, however, this is not considered a problem as the difference in daily solar radiation is small between the two locations that are within short distance of each other in this case approximately 50 km. The solar radiation profile is aggregated to monthly average values.

4. The stochastic formulation

By formulating the model using stochastic differential equation will enable the modelling of uncertainties of the entire model. With the correct noise formulation the parameters of the model can be determined through

maximum likelihood principles. This principle in short is to look at the discrepancies between the measurements and the model output. By minimizing the mean squared error of the output error (simulatation error) the parameter values can be estimated. This principle has been used with great success in (Madsen, 2000, 2003). However, if the maximum likelihood is the goal then these methods are not sufficient. In order to get maximum likelihood estimates the model has to be transformed into a stochastic formulation. In this case, since the model is continuous a stochastic differential equation interpretation (SDE) is chosen. The general framework of the SDE can in short be written on continuous discrete states space form as

$$d\mathbf{x}_t = f(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\theta}_t)dt + g(\mathbf{u}_t, \boldsymbol{\theta}_t)d\boldsymbol{\omega}_t \tag{9}$$

$$\mathbf{y}_{t_k} = h(\mathbf{u}_{t_k}, \boldsymbol{\theta}_{t_k}, \mathbf{x}_{t_k}) + \mathbf{e}_{t_k}, \tag{10}$$

where f is the drift term and $g(\mathbf{u}_t, \boldsymbol{\theta}_t)$ is the diffusion of the process. The function g is dependent on parameters $\boldsymbol{\theta}$ and the input $\mathbf{u}$. In the general case the diffusion term could also be dependent on the state $\mathbf{x}$, however, this can be difficult and is beyond the scope of this paper. $d\boldsymbol{\omega}$ is the standard Wiener process. The observation equation (10) is the mapping from the model space to observation space, with observation noise $\mathbf{e}_{t_k}$.

By relating the system noise through the function g in equation (9) will not only give information for the estimation later on, it will also help reduce the measurement error in (10) since much of the noise can be correlated to inputs, model approximations and parameters. In short the benefits of the continuous-discrete stochastic state space formulation (9) - (10) are partly originating from the fact that the diffusion part of the SDE accounts for: (Madsen and Holst, 1995)

- Modeling approximations. For instances the dynamics, as described by the model operator f in eq. (9), might be an approximation to the true system.
- Unrecognized and unmodeled inputs. Some variables which are not considered.
- Measurements of the input are noise-corrupted. In this case the measured input is regarded as the actual input to the system, and the deviation from the true input contributes to $d\boldsymbol{\omega}_t$

We will show that the SDE formulation will enable the Maximum Likelihood estimator (MLE). The MLE will facilitate an estimation of every parameter in the system, including model noise, input noise and observation noise. With the given stochastic model (9) the MLE of the unknown parameters can be obtained by finding the parameters $\boldsymbol{\theta}$ that will maximize the likelihood function of a given sequence of observations $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k, \ldots, \mathbf{y}_N$. The following notation will be used $\mathcal{Y}_k = [\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{y}_0]$. The maximum likehood is given as the joint density of all observations, i.e.

$$L(\boldsymbol{\theta}; \mathcal{Y}_T) = p(\mathcal{Y}_T|\boldsymbol{\theta}), \tag{11}$$

where $\mathcal{Y}_T$ is the sequence of measurements given at time points $t \in \{1, \ldots, T\}$, $\boldsymbol{\theta}$ is the unknown parameters that maximizes the likelihood function. Using $P(A \cap B) = P(A|B)P(B)$, the likelihood function can be written as a product of conditional probability densities,

$$L(\boldsymbol{\theta}; \mathcal{Y}_T) = \left(\prod_{t=1}^{T} p(\mathbf{y}_t|\mathcal{Y}_{t-1}, \boldsymbol{\theta}) \right) p(\mathbf{y}_0|\boldsymbol{\theta}). \tag{12}$$

In order to facilitate the MLE the $p(\mathbf{y}_0|\boldsymbol{\theta})$ must be known and the sequence of all conditional densities must be found by solving the Fokker-Planck equation. However, this approach is computational infeasiable in pratice. The diffusion term is not depended on the state variable and therefore the iterative extended Kalman filter Jazwinski (1970) can be used to find the conditional densities. However, It can be put into a more practical setting by introducing the one-step prediction, the measurement covariance, and the prediction error,

$$\begin{aligned} \hat{\mathbf{y}}_{t|t-1} &= E\{\mathbf{y}_t|\mathcal{Y}_{t-1}, \boldsymbol{\theta}\} \\ \mathbf{R}_{t|t-1} &= V\{\mathbf{y}_t|\mathcal{Y}_{t-1}, \boldsymbol{\theta}\} \\ \boldsymbol{\varepsilon}_t &= \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1} \end{aligned} \tag{13}$$

and assuming that the conditional probability densities are Gaussian (12) the likelihood function can be rewritten as

$$L(\boldsymbol{\theta}; \mathcal{Y}_T) = \left(\prod_{t=1}^{T} \frac{\exp\left(-\frac{1}{2}\boldsymbol{\varepsilon}_t^T \mathbf{R}_{t|t-1}^{-1} \boldsymbol{\varepsilon}_t\right)}{\sqrt{\det\left(\mathbf{R}_{t|t-1}\right)\sqrt{2\pi}^n}} \right) p(\mathbf{y}_0|\boldsymbol{\theta}),$$

where $\mathbf{R}_{t|t-1}$ and $\boldsymbol{\varepsilon}_t$ can be evaluated using the extended Kalman Filter for non-linear models. n denotes the number of outputs in the vector $\mathbf{y}_t$. The

maximum likelihood estimate for θ is obtained by maximizing the likelihood function. This is equivalent to minimizing the following expression by taking the negative logarithm of the expression, thus

$$-\ln\left(L(\boldsymbol{\theta};\mathcal{Y}_T|\mathbf{y}_0)\right) = \frac{1}{2}\sum_{t=1}^{T}\left(\ln\left(\det\left(\mathbf{R}_{t|t-1}\right)\right) + \boldsymbol{\varepsilon}_t^T\mathbf{R}_{t|t-1}^{-1}\varepsilon_t\right) n\ln\left(2\pi\right) \quad (14)$$

and then solving the nonlinear optimization problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta}\{-\ln\left(L(\boldsymbol{\theta};\mathcal{Y}_T|\mathbf{y}_0)\right)\}, \quad (15)$$

where Θ is the parameter space.

5. CTSM

In this paper a semi-automatic tool will be used to calibrate the parameters of the simple eco system model. This automatic tool is the very well documented Continuous Time Stochastic Modelling (CTSM) (Kristensen et al., 2004). CTSM is a tool that can estimate parameters in a given model as long as the model is formulated as an SDE driven continuous-discrete time state space model. There is a limitation to the diffusion term namely that it has to be independent of the state vector. However, in this setting of estimating the parameters for the simple eco model it has no real consequence. There are other semi automatic toolboxes that could be used e.g. MoCaVa (Bohlin, 2001) which is tool box for matlab, however, CTSM has to be very reliable and easy to use (Kristensen et al., 2004). CTSM is platform independent and shareware. It can be downloaded from (`http://www2.imm.dtu.dk/~ctsm/`). To utilize the benefits described in the previous section the model (5) - (7) has to undergo a transformation from a ODE into a SDEs, using,

$$d\mathbf{C} = \frac{d\mathbf{C}'}{dt'}dt + \mathbf{g}(\mathbf{C}',\boldsymbol{\theta})d\omega, \quad (16)$$

where $\mathbf{C}'$ is the concentration of the biogeochemical species given in (5) - (7). The $\mathbf{C}$ and $\boldsymbol{\theta}$ are states and parameters of the stochastic process. The diffusion term is given by $\mathbf{g}(\mathbf{C}',\boldsymbol{\theta})$ and the term $d\boldsymbol{\omega}$ is the Wiener process. In this paper the diffusion term will be regarded as constant, $\boldsymbol{\sigma}$ and, hence the final model considered using CTSM is

$$d\mathbf{C} = \frac{d\mathbf{C}'}{dt'}dt + \boldsymbol{\sigma}d\boldsymbol{\omega} \quad (17)$$

6. Results

The initial guess for the parameters was taken from the default values in ECOLAB and is given in Table 1.

K_I	$K_{I_{PN}}$	$K_{I_{DN}}$	K_{IN}	K_s	K_{DN}	K_{PN}	θ_s	z
10.0	20.0	0.01667	0.05	0.5	0.01	0.3	1.04	20.0
K_m	θ_m	K_D	θ_d	μ_{max}	θ_{pp}	RAC	FAC	
0.5	1.04	0.01	1.04	1.2	1.04	1.3	0.5	

Table 1: The initial parameter values from ECOLAB.

6.1. Results from CTSM

The CTSM had no difficulties in estimating the parameters in the lumped model. However, it took some iteration to find the final parameter values. The parameter values where all chosen from Table 1 but an modified such that a CTSM would perform a estimation run. Due to the small data set not all parameters could be estimated. Therefore we have chosen to fix some parameters in the estimation. First we will only try to estimate the initial value for PN and IN in the model. The initial value for DN was chosen to be fixed at 0.88. We have also chosen to fix μ_{max} and θ_d and the fixed values and the other values and bounds can be seen in the Table 3.

The performance of the estimation can be seen in Table 2, 3, and 4. CTSM has many output parameters that will be helpful in the calibration. In the first optimization shown in Table 2 the values from the objective function, the penalty function and the negative logarithm of determinant of the Hessian matrix of the objective function are given, along with the number of iterations and the number of object evaluations of the optimization. Subtracting the penalty function from the objective function yields the negative logarithm of likelihood function. The value of the negative inverse of the Hessian matrix should be as small as possible, because it is the covariance estimation of the parameters in the model. Table 2 shows the estimated parameter values. The first five columns are inputs to the estimation. The first two columns gives the name of the parameter and whether it is fixed or free (ML). The next three columns are the lower and upper bounds of the parameter along with the initial values. The next six columns are output from the optimization. The first column is the estimated value and in the second column is the standard deviation of the estimated parameter. The next two columns are the t-score and the p($>$ |t|) value is the fraction of probability of the corresponding

t-distribution outside the limits under the hypothesis that the parameter is zero set by the t-score. If this value close to 1 then the parameter could have been left out of the model. The two last columns are reserved for the gradient of the objective function and the gradient of the penalty function with respect to the given parameter. The gradient of the objective function most be close to zero which would indicate that the optimization is close to a (local) optimum of the objective function. The gradient of the penalty function most also be close to zero; otherwise if the gradient is negative then it indicates that the lower constraint of the parameter is too narrow and has to be loosen and vice versa it the gradient is positive.

The final estimation shown in Tables 2, 3, and 4 are the product of many calibration runs with CTSM. There is much work in finding the right initial values. The initial values from ECOLAB were not usable because the simpler model lacks much of the dynamics of ECOLAB. ECOLAB is coupled to a three dimensional hydrological model that provides the hydrodynamics of the lake. There is also various other input to ECOLAB that has be eliminated in the simple model. However, the way to go about the calibration in CTSM is to start with a small number of free parameters that CTSM has to estimate. All other parameters are being fixed at values that we have perturbed from the initial values in order to get CTSM to run properly. After the initial estimation the estimates are checked for the gradient of respectively the objective and penalty function. If the values are large then the constraints must be loosen. CTSM is run again and if the estimates are reasonable the other parameters are being set free in the CTSM. In this way the final estimates are derived.

In Table 2 the values of the objective and penalty functions along with the negative logarithm of the determinant of the Hessian matrix are given. These values are not of importance in this paper, however, in other calibrations studies these values can tell if the model is over parameterized or not. Here Table 2 is just shown as a part of CTSM output options. In Table 3 the final estimation values are shown. Every parameter in the model has been estimated through the maximum likelihood setting. The values from the gradient of the penalty function are almost all close to zero. There are, however, the two variance parameter for the mean error, $s_1 = -5.67$ and $s_2 = -1.71$ that are not close to zero. We tried to loosen the bounds on these two parameters, however, the estimation became unstable and the estimation could not be completed. Therefore we had to fixed the bounds to the values in Table 3. For these two values the gradient of the objective function is very

close to zero and therefore it is assumed that the small offset in the gradient of the penalty function is neglectable. The reason behind this is of course the small data set. If we have had a large data set the observation errors could be better estimated and thereby the penalty function could have gone to zero. The values of the gradient of the objective function are also close to zero which indicates that a (local) maximum is found.

However, due to the small data set at hand for the estimation; we get fairly large standard deviations for most of the parameter values. There are also large values in the probability p($>$ |t|) which indicates that most of the parameters has little significance and could be left out of the model. However, before we discard the estimation we must take into account that our data set is very small for an estimation of 19 parameters. In order to decrease the standard deviation and the t-probability more parameters in the model could be fixed. We have tried to do this in later estimation runs and found that we could decrease these values. However, simulations where we have fix many of the parameters did not give as good results when we plotted the model versus the observations. The figures showed that the model did not catch the dynamics of the observations. Therefore we will accept the large standard deviation in the estimation. We can only decrease these values if we had a large data set. However, since the measuring of this particular site is closed down we can not hope to get more data from this site.

Finally Table 4 shows the correlation Table of the parameters. There are no high correlations in the estimation. The correlation is only a problem if the correlations are close to 1.

Value of objective function	-1.925719349534588E+00
Value of penalty function	5.653813512599075E-02
Negative logarithm of determinant of Hessian	-8.274370174186501E+00
Number of iterations	253
Number of objective function evaluations	325

Table 2: Optimisation Results

Name		Min. value	Initial value	Max. value	Prior std. dev.	Estimate	Std. dev.	t-score	p(¿—t—)	dF/dPar	dPen/dPar
PN_0	ML	0.1	0.25	1.0	N/A	3.1734E-01	2.3591E-01	1.3452	0.4648	0.0000	0.0000
DN_0	Fix	N/A	0.88	N/A	N/A	0.88	N/A	N/A	N/A	N/A	N/A
IN_0	ML	0.3	0.43	2.0	N/A	4.8038E-01	5.0668E-01	0.9481	0.5548	0.0000	-0.0004
$K_{I_{PN}}$	ML	1.0	25.0	30.0	N/A	1.0365E+00	4.5133E-01	2.2966	0.3664	-0.0000	-0.0778
$K_{I_{DN}}$	ML	0.01	0.05	2.0	N/A	1.9622E+00	3.3921E-01	5.7847	0.3029	0.0000	0.2749
K_I	ML	0.0010	10.0	120.5	N/A	5.6109E+01	5.6487E+01	0.9933	0.5421	-0.0000	0.0002
K_{IN}	ML	1.0E-8	0.05	1.0	N/A	2.5134E-06	2.3635E-03	0.0011	0.9994	0.0000	-0.0000
K_s	ML	1.0E-4	0.5	1.5	N/A	2.5695E-04	1.3581E-02	0.0189	0.9887	0.0000	-0.0001
K_{DN}	ML	1.0E-4	0.05	1.0	N/A	1.7537E-04	4.5083E-03	0.0389	0.9767	0.0000	-0.0003
K_{PN}	ML	1.0E-7	1.0E-4	1.0	N/A	3.9912E-01	3.6687E-01	1.0879	0.5178	-0.0000	0.0001
θ_s	ML	0.9	1.04	1.8	N/A	1.2886E+00	1.4192E+01	0.0908	0.9458	0.0000	0.0001
K_m	ML	1.0E-5	1.0E-4	1.0	N/A	9.7930E-01	1.9629E-01	4.9889	0.3075	0.0000	0.2285
θ_m	ML	1.0	1.04	1.2	N/A	1.1595E+00	1.4801E-01	7.8341	0.2966	-0.0000	0.0804
K_D	ML	1.0E-4	0.0010	1.0	N/A	2.8790E-04	1.7536E-02	0.0164	0.9902	0.0000	-0.0001
θ_d	Fix	N/A	1.04	N/A	N/A	1.04	N/A	N/A	N/A	N/A	N/A
μ_{max}	Fix	N/A	1.2	N/A	N/A	1.2	N/A	N/A	N/A	N/A	N/A
θ_{pp}	ML	1.0	1.04	1.8	N/A	1.0657E+00	2.0384E-01	5.2280	0.3059	-0.0000	-0.0243
σ_1	ML	0.0010	1.0	10.0	N/A	1.7023E-03	3.6828E-02	0.0462	0.9724	-0.0000	-0.0003
σ_2	ML	0.0010	1.0	10.0	N/A	1.3645E-03	1.4196E-02	0.0961	0.9427	0.0000	-0.0010
σ_3	ML	0.0010	1.0	10.0	N/A	1.3590E-03	1.3765E-02	0.0987	0.9411	-0.0000	-0.0011
ε_1	ML	0.1	1.0	2.0	N/A	1.0042E-01	1.8436E-03	54.4710	0.2890	0.0000	-5.6714
ε_2	ML	0.1	1.0	2.0	N/A	1.0077E-01	5.4451E-03	18.5056	0.2902	-0.0000	-1.7195

Table 3: Estimation Results

	PN_0	IN_0	$K_{I_{PN}}$	$K_{I_{DN}}$	K_I	K_{IN}	K_s	K_{DN}	K_{PN}	θ_s	K_m	θ_m	K_D	θ_{pp}	σ_1	σ_1	σ_3	ε_1	ε_2
PN_0	1																		
IN_0	0.0727	1																	
$K_{I_{PN}}$	-0.0151	-0.0009	1																
$K_{I_{DN}}$	0.0515	0.0265	-0.0569	1															
K_I	-0.1592	-0.5802	-0.1034	-0.0600	1														
K_{IN}	0.0638	-0.1039	-0.1181	0.0759	0.2038	1													
K_s	0.0266	-0.0026	-0.0137	0.0212	0.0163	-0.0005	1												
K_{DN}	0.0195	0.0119	-0.0226	0.0204	0.0039	0.0982	0.0600	1											
K_{PN}	0.1923	0.7920	0.0175	0.0276	-0.4867	-0.1264	-0.0132	-0.0392	1										
θ_s	0.1190	0.0872	-0.0638	0.0302	-0.1362	0.7314	0.1272	0.2157	0.0024	1									
K_m	0.0103	-0.0056	0.0060	-0.0395	0.0557	0.0236	0.0198	-0.0001	0.0008	-0.0228	1								
θ_m	0.1242	0.8313	-0.0103	0.0471	-0.7606	-0.1535	-0.0124	0.0188	0.6486	0.0962	0.0922	1							
K_D	0.0153	-0.0251	-0.0154	0.0056	0.0405	-0.0968	-0.0090	0.0086	-0.0339	0.0553	-0.0031	-0.0429	1						
θ_{pp}	0.0391	-0.6275	0.0147	-0.0704	-0.2179	-0.0789	-0.0310	0.0389	-0.4864	0.0081	-0.0346	-0.2738	-0.0099	1					
σ_1	-0.0074	0.0172	0.0513	-0.0321	-0.0256	-0.0602	-0.0380	-0.0121	0.0079	-0.2326	0.0061	0.0283	-0.0126	0.0049	1				
σ_2	0.0197	0.0239	0.0391	0.0203	0.0010	0.0160	0.0101	0.0299	0.0067	0.1519	-0.0090	-0.0043	-0.0053	-0.0309	0.0160	1			
σ_3	-0.0119	0.0330	0.0123	-0.0242	-0.0225	-0.0043	-0.0107	-0.0057	0.0373	-0.0328	0.0053	0.0397	0.0221	-0.0189	0.0434	-0.0391	1		
ε_1	0.0481	0.0118	-0.0095	0.0471	-0.0052	0.1190	0.0327	0.0487	-0.0213	0.1943	-0.0120	0.0090	0.0076	-0.0190	-0.0569	0.0275	-0.0234	1	
ε_2	-0.0671	-0.0199	0.0666	-0.0304	0.0203	-0.1682	-0.0317	-0.1021	0.0147	-0.2836	0.0138	-0.0094	0.0115	0.0023	0.1181	-0.0585	0.0194	-0.1709	1

Table 4: Correlation Matrix

The final test is a simulation with the highly simplified and newly estimated model and a comparison with the measure time series. We have chosen to show the model time series as ODE solutions since the size of the diffusion coefficients are very small $(\sigma_1, \sigma_2, \sigma_3) = (1.6236E-03, 1.4221E-03, 1.3597E-03)$. The ODE solution can be considered the average SDE solution over many trajectories. In Figure 5 the simulation of PN is shown together with the observations of PN. The model captures some of the dynamics of the observations; however, we can still see that there is some autocorrelation. In Figure 6 a simulation of DN is shown. The dynamics of the

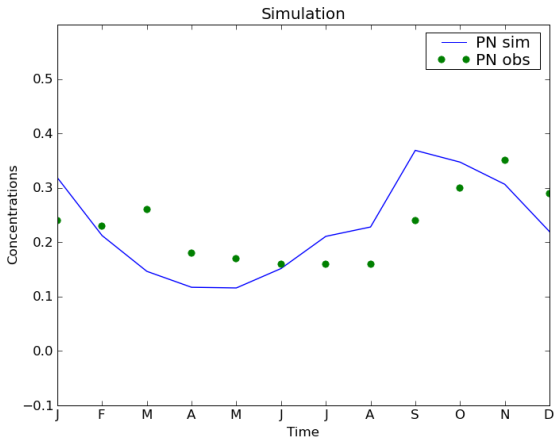


Figure 5: The simulation of PN with the simplified model versus the observations. The model captures some of the observed dynamics

model and observation from other lakes suggests that this is a typical dynamic behavior of DN. However, due the lack of observations for the last months of the year the DN concentrations goes towards zero. This is of course not desireable, however, if more data were available this behavior would probably not arise. In Figure 7 the IN component is shown together with the measurements. Here we only had eight measurement points from April to November. However, the dynamic of the observations is very well captured even though we have to suspect fairly large measurement errors.

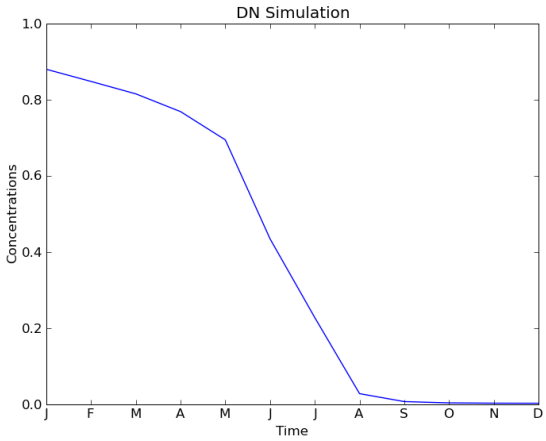


Figure 6: The simulation DN with the estimated model versus the observations. The model captures some of the observation dynamics

7. Conclusion

An estimation or calibration of the highly simplified model was archived with a stochastic formulation of the lumped model. It was shown that by introducing a stochastic diffusion term in the model parameters could be estimated through the maximum likelihood theory by identifying the model and measurement errors. Reasonable parameter estimation results where obtained through the use of the freeware software CTSM which is able to estimate embedded parameters in stochastic state space models with discrete observations. The model parameters were all estimated within the constraints of CTSM and the constraints set in this paper. The estimation was constraint by the small data set that we had at our disposal. This small data set is reflected in the large standard deviation for most of the parameters and in the probabilities of the student t-score. However, as already argued these problems can be reduced with a larger data set. The size of the data set in this paper is, however, typical of a lake eco system model. The measuring campaign at Esrum lake is closed down and we cannot expect to get a better data set with more data points. However, within the constraints of this data set we were still able to estimate or calibrate the model parame-

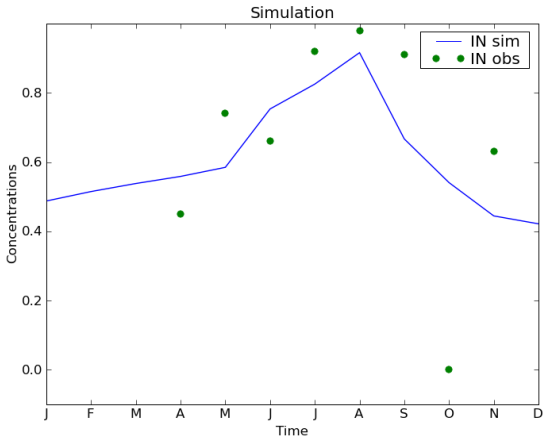


Figure 7: The simulation IN with the estimated model versus the observations. The model captures some of the observation dynamics

ters such that we could mimic the observed dynamics of the biogeochemical species measured in the lake.

Acknowledgements

This work has been financially supported by the Danish Research School ITMAN. Data was provided through the CONWAY project granted by Danish Research Council (J. nr. 2055-01-0034)

References

Bohlin, T. (2001). A grey box process identification tool: Theory and practice. Technical report, Royal Institute of Technology, Stockholm.

Bohlin, T. (2006). *Practical Grey-box Process Identification: Theory and Applications (Advances in Industrial Control)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Bohlin, T. and S. F. Graebe (1995). Issues in nonlinear stochastic grey box identification. *International Journal of Adaptive Control and Signal Processing 9*, 465–490.

Frederiksborg Amt (2001). Esrum sø - tilstand og udvikling. Technical report, Vandmiljøovervågning nr. 74, Frederiksborg Amt, Denmark pp. 79.

Frederiksborg Amt (2002). Esrum sø - tilstand og udvikling, supplement. Technical report, Vandmiljøovervågning, Frederiksborg Amt, Denmark pp. 33.

Gardiner, C. (2004). *Handbook of Stochastic Methods* (THIRD ed.). Springer Verlag.

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory* (FIRST ed.). Dover Publications INC.

Kristensen, N. R., H. Madsen, and S. B. Jørgensen (2004). Parameter estimation in stochastic grey-box models. *Automatica 40*, 225–237.

Lessin, G. and U. Raudsepp (2006). Water quality assessment using integrated modeling and monitoring in narva bay, gulf of finland. *Environmental Modeling & Assessment 11*, 315–332'.

Madsen, H. (2000). Automatic calibration of a conceptual rainfall-runoff model using multiple objectives. *Journal of hydrology 235*, 276–288.

Madsen, H. (2003). Parameter estimation in distributed hydrological catchment modelling using automatic calibration with multiple objectives. *Advances in Water Resources 26*, 205–216.

Madsen, H. and J. Holst (1995). Estimation of continuous-time models for the heat dynamics of a building. *Energy and Building 22*, 67–79.

Madsen, H. and H. Melgaard (1991). The mathematical and numerical methods used in ctlsm - a program for ml-estimation in stochastic, continuous time dynamical models. Technical report, Research Report No. 7/1991, IMSOR, 22 pp, Technical University of Denmark, Lyngby, Denmark.

Melgaard, H. and H. Madsen (1991). Ctlsm – continuous time linear stochastic modelling. Technical report, Research Report No. 1/1993, IMSOR, 53 pp, Technical University of Denmark, Lyngby, Denmark.

Øksendal, B. (2005). *Stochastic Differential Equations* (6th ed.). Springer Verlag.

Young, P. (1981). Parameter estimation for continuous-time models - a survey. *Automatica 17*, 23–39.

APPENDIX C

Paper C

Accepted for publication in Atmospheric Chemistry and Physics

This manuscript was prepared for Atmos. Chem. Phys. (Discuss.)
with version 7.7 of the egu LaTeX class egu.cls.
Date: July 20, 2009

ACP-LOGO

Implementation and testing of a simple data assimilation algorithm in the regional air pollution forecast model, DEOM

J. Frydendall[1], **J. Brandt**[2], **and J. H. Christensen**[2]

[1]DTU Informatic, Technical University of Denmark, Denmark
[2]National Environmental Research Institute, Aarhus University, Denmark

Abstract. A simple data assimilation algorithm based on statistical interpolation has been developed and coupled to a long-range chemistry transport model, the Danish Eulerian Operational Model (DEOM), applied for air pollution forecasting at the National Environmental Research Institute (NERI), Denmark. In this paper, the algorithm and the results from experiments designed to find the optimal setup of the algorithm are described. The algorithm has been developed and optimized via eight different experiments where the results from different model setups have been tested against measurements from the EMEP (European Monitoring and Evaluation Programme) network covering a half-year period, April-September 1999. The best performing setup of the data assimilation algorithm for surface ozone concentrations has been found, including the combination of determining the covariances using the Hollingsworth method, varying the correlation length according to the number of adjacent observation stations and applying the assimilation routine at three successive hours during the morning. Improvements in the correlation coefficient in the range of 0.1 to 0.21 between the results from the reference and the optimal configuration of the data assimilation algorithm, were found. The data assimilation algorithm will in the future be used in the operational THOR integrated air pollution forecast system, which includes the DEOM.

1 Introduction

Even though the field of chemical weather forecasting is still very much in the research and development phase, operational forecasting of the air pollution concentration is now being carried out on a routine basis in many countries throughout the world. The chemical weather can be seen as analogous to the meteorological weather. In particular, chemical weather emphasizes the strong influence of meteorological variability - and the chemical response to this variability - on air quality (Lawrence et al., 2005). In contrast to numerical weather forecasting, it is technically possible to carry out operational chemical weather forecasting without using data assimilation of the prognostic variables in the air pollution model. Without data assimilation of meteorological parameters during initialization, numerical weather forecast models would produce simulations that - even though the results would appear realistic - have nothing to do with the actual weather. A long-range chemistry-transport model (CTM) used for operational forecasting is driven by a numerical weather forecast model, and is bound by a emissions inventory as well as chemical lifetimes of the individual species. In this way the results from a chemical weather forecast will show good performance when compared against measurements. However, applying the data assimilation techniques which have been used by the weather forecasting community for since the late 1950ties (Gandin, 1963), has the potential to make significant improvements in chemical weather forecasts and these techniques are now being introduced in air pollution models by various scientific communities.

At the National Center for Atmospheric Research, Atmospheric Chemistry Division, Boulder, USA an Optimum Interpolation routine (Lamarque et al., 1999) is being used to investigate CO in the troposphere. A group at the Data Assimilation Office, National Aeronautics & Space Administration (NASA) Goddard Space Flight Center USA, has used a Kalman Filter (Ménard and Chang, 2000; Ménard et al., 2000) to investigate chemical tracers. In Europe there are several groups working with chemical data assimilation: At the University of Cologne, Germany, a four-dimensional variational algorithm for atmospheric chemistry modelling has been developed and used in the EURopean Air Pollution Dispersion (EURAD) model (Elbern et al., 1997; Elbern and Schmidt, 1999; Elbern et al., 2000). At the Delft University

Correspondence to: J. Frydendall (jf@imm.dtu.dk)

of Technology, Netherlands, a Kalman Filter has been developed (van Loon and Heemink, 1997) for atmospheric chemistry modelling. At the French meteorology laboratory, an Optimum Interpolation routine for ozone analysis has been developed (Blond et al., 2003; Blond and Vautard, 2004). At Norwegian Institute for Air Research, (NILU), a number of statistical interpolation methods are being employed for PM_{10} and applied in the Unified EMEP model (Denby et al., 2008). In the Swedish Meteorological and Hydrological Institute (SMHI), an operational 2D-var method is under development for operation in the Multiscale Atmospheric Transport and Chemistry Model (MATCH) model (Denby et al., 2008). At NERI, Denmark, besides of the work described in this paper, development and tests of a four-dimensional variational method have been made, see Zlatev and Brandt (2005, 2007).

Chemical transport models are valuable tools to understand the transport of chemical pollutants in the atmosphere. However, due to uncertainties e.g. caused by discretization of the governing equation, uncertainties in the simplified chemical reaction scheme or physical parameterizations, or erroneous emissions, the CTMs cannot truly represent the real world. On the other hand uncertainties in the measurements makes the comparison between the CTMs and the observations a non-trivial business. Data assimilation routines combines the information from the CTMs and the measurements by taking into account the model and observation uncertainties to make better representation of the air pollution fields.

A general problem in chemical data assimilation is, however, the lack of real-time data. In the meteorological community, a dense network for real-time meteorological measurements, both at the surface as well as radio soundings, was established many years ago. With respect to chemical data assimilation, the research groups typically have to collect the available sparse data sets on their own. However, more and more real-time surface observations are becoming available for assimilation, and even satellite measurements of e.g. the tropospheric column of NO_2 can be obtained. Potentially, these data sets together provide relative high accuracy from the surface measurements combined with the greater spatial coverage from the satellite data. However, none of them provides an estimate of the vertical distribution of the chemical species.

An alternative to the direct use of data assimilation is postprocessing approaches. In these postprocessing approaches, a moving training window is assign to a fixed number of days where the model uncertainties are estimated from error residuals between model forcasts and observations. The model uncertainties estimates are used as bias corrections in the future forecast window. A nice example of such a postprocessing techniques is demonstraited in (Kang et al., 2008).

Data assimilation techniques applied in chemistry-transport models cannot only be used for operational forecasting of the chemical weather but also for generating analyzed fields covering a large time period of the different air pollution species e.g. for monitoring the air quality and assessing the impacts from air pollution. Examples could be integrated monitoring (using both models and measurements, see e.g. Hertel et al. (2007)) of nitrogen species with respect to eutrophication in the marine and terrestrial ecosystems or integrated monitoring of ozone, nitrogen-oxides and particulate matter with respect to the impacts on human health.

2 The DEOM model

The long-range chemical transport model, the Danish Eulerian Operational Model (DEOM) (Brandt et al., 2000, 2001a,b,c) has been developed at NERI for air quality forecasting. The model includes emissions, atmospheric transport and dispersion, chemical transformations and dry and wet depositions of 35 chemical species. The domain of the DEOM covers Europe and is constructed so that it is covered by the domain of the meteorological model, Eta, applied for operational weather forecasting at NERI and used as a driver for the DEOM. The Eta model is discretizised on a staggered latitude/longitude system with shifted pole. The horizontal grid resolution is 0.25 $^\circ$ $\times$ 0.25 $^\circ$ corresponding to approximately 39 km $\times$ 39 km at 60 $^\circ$ N. The number of horizontal grid points is 104 $\times$ 175 and the number of vertical layers is 32. The DEOM model is applied on a polar stereographic projection. The horizontal grid resolution is 50 km $\times$ 50 km at 60 $^\circ$ N. The number of grid points is 96 $\times$ 96. Three vertical layers are used in the DEOM model. The three layers are defined as a mixed layer (below the mixing height), a smog or reservoir layer between the mixing height and the advected mixing height from the previous day. The top layer is located between the advected mixing height and the free troposphere. The model has been a part of various inter-comparison studies and has shown comparable results with similar models, see e.g. Tilmes et al. (2002).

A splitting procedure, based on the ideas of McRae et al. (1982), is applied in the DEOM. The horizontal transport is discretizised using an accurate space derivative algorithm. Time integration is performed with a predictor corrector scheme with several correctors. For the horizontal dispersion, truncated Fourier series approximate the concentrations. Dry and wet depositions are computed directly using simple parameterizations. The chemical scheme used in the model is the CBM-IV scheme with 35 species. Chemistry is solved using the QSSA method (Hesstvedt et al., 1978).

The DEOM model is a part of the THOR integrated model system, Brandt et al. (2001a,b,c, 2005), capable of performing forecasting of meteorological and chemical weather for the general public as well as assessment and management for decision-makers in general. The system consists of several meteorological and air pollution models, developed at NERI over recent decades, and is capable of operating for different applications and at different scales. Global meteorological data from NCEP are used as initial and bound-

ary conditions for the numerical weather forecast model Eta. The weather data from this model are used to drive the air pollution models: the Danish Eulerian Operational Model (DEOM), the Urban Background Model (UBM), the Operational Street Pollution Model (OSPM) and others. Air pollution data from the DEOM is used as input to the UBM and the results from this model is used as input to the OSPM, see Brandt et al. (2001c).

Coupling models over different scales makes it possible to account for contributions from local, near-local as well as remote emission sources in order to describe the air quality at a specific location - e.g. in a street canyon or in a suburban area. The system provides high-resolution three-day forecasting of weather and air pollution, from regional scale over urban background scale and down to individual street canyons in cities - on both sides of the streets. The whole system is run operationally and automatically four times every day, initiated at 00 UTC, 06 UTC, 12 UTC and 18 UTC. The system is also applied in connection with the urban and rural monitoring programs in Denmark where the model results and measurements are used together via integrated monitoring to obtain the best available information level for the atmospheric environment and effects. It is planned that the data assimilation routine developed in this study is to be used as a basis for improvements in the air quality forecast at regional scale, which will also affect the results on urban scales.

3 The data assimilation algorithm

The data assimilation algorithm in this article is the based on a Statistical Interpolation algorithm. The notation used is similar to the notation introduced by Ide et al. (1997). The observations $\boldsymbol{y}_o$ represent a measure of the real world. The data assimilation algorithm introduces this knowledge into the model and the combination of the model state $\boldsymbol{x}_b$ and observation state $\boldsymbol{y}_o$ is called the analysis state $\boldsymbol{x}_a$ which in theory should be a better representation of the real world than the background state or the observation state individually. The analysis state is obtained by weighting the model errors against the observation errors. This leads to the interpolation equation (Bouttier and Courtier., 1998):

$$\boldsymbol{x}_a = \boldsymbol{x}_b + \mathbf{K}\left(\boldsymbol{y_o} - \mathbf{H}\left(\boldsymbol{x}_b\right)\right) \tag{1}$$

$$\mathbf{K} = \mathbf{B}\mathbf{H}^T\left(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{2}$$

where the linear operator $\mathbf{K}$ is called the Kalman gain and is the weight matrix of the analysis. $\mathbf{H}$ denotes the linear map between model space and observation space.

4 The background error covariance matrices

Three different background error covariance matrices $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3)$ will be tested and compared to each other. It is assumed that the horizontal correlation is homogeneous and isotropic for the two first background error covariance matrices. For the last background error covariance matrix it is only assumed that the horizontal correlation is homogeneous. The first background error covariance matrix $\mathbf{B}_1$ is the well known scaled Balgovind function (Balgovind et al., 1983),

$$C_0(\boldsymbol{r}) = \left(1 + \frac{|\boldsymbol{r}|}{L}\right)\exp\left(-\frac{|\boldsymbol{r}|}{L}\right) \tag{3}$$

where $\boldsymbol{r}$ is Euclidean distance between the grid cell locations and L is the correlation length. For a thorough review on the properties of the Balgovind function and other correlation functions, see Gaspari and Cohn (1999).

The second background error covariance matrix $\mathbf{B}_2$ is defined by Hoelzemann et al. (2001). The background error covariance matrix takes into account that adjacent observation stations can deteriorate the analysis field. The function is defined as follows: Let δ be the number of observation stations neighboring a model grid point, for which the radius of influence has to be estimated. The more adjacent the observation station is to the model grid point, the smaller the radius of influence. Taking 20% as the lower limit, the new L becomes

$$\tilde{L}\left(\delta\right) = \left(1 - \frac{\delta}{10}\right)L \tag{4}$$

where $0 \leq \delta \leq 8$. The new correlation matrix becomes

$$C_0(\boldsymbol{r}) = \left(1 + \frac{|\boldsymbol{r}|}{\tilde{L}}\right)\exp\left(-\frac{|\boldsymbol{r}|}{\tilde{L}}\right) \tag{5}$$

Finally the last background error covariance matrix $\mathbf{B}_3$ takes into account that the observation spreading done by the background error covariance matrix should depend on the wind direction and the wind speed. With this approach the assumption on horizontal isotropic characteristic is abandoned in order to get a more realistic correlation function. The correlation length is decomposed into two correlation lengths: One that is parallel with the wind direction and one that is perpendicular to the wind direction, that is $L \rightarrow L_{\parallel} + L_{\perp}$. The isotropic correlation function can be interpreted as correlation circle in a 2-D system where the correlation length is the radius of the correlation circle. In the anisotropic case, the correlation circle will be transformed into a correlation ellipse with the major and minor axis given as a function of the correlation lengths in the wind directions.

Given the wind $V = V(u, v)$, we can calculate the rotations matrix and the transformation of (x, y):

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x\cos\varphi + y\sin\varphi \\ -x\sin\varphi + y\cos\varphi \end{pmatrix} \tag{6}$$

where $\varphi = v/u$. Hence $x \rightarrow \frac{x'}{L_{\parallel}}$and $y \rightarrow \frac{y'}{L_{\perp}}$. The magnitude of the correlation lengths can be determined in several ways. In this case we let the magnitude be a function of the

ratio between the wind components:

$$\frac{L_\perp}{L_\parallel} = v/u$$
$$L_\parallel = L \tag{7}$$

Then the background correlation function becomes

$$C_0(r') = (1 + r')\exp(-r') \tag{8}$$

where $r' = (x'^2 + y'^2)^{1/2}$. The adjacent function (4) can, of course, be combined with (8).

4.1 Covariance determination

An essential task in data assimilation is the estimation of the error parameters of the model and the observations. The approach chosen in this paper is called the Hollingsworth method (Hollingsworth and Lonnberg, 1986; Lonnberg and Hollingsworth, 1986; Daley, 1996). The idea is to look at the auto correlation function of the residuals between the model forecast and the observations. The sample correlations among all pairs of stations can be plotted as a function of separation distances, together with a curve representing a fitted auto correlation model, cf eq. (3). By extrapolation of the curve to the origin, the ratio between the observation and forecast error standard deviations can be determined. Another commonly used approach, which is used for estimating parameters in a very large state space model is based on the Ensemble Kalman Filter (Evensen, 1994; Burges et al., 1998), and this approach will be tested in the future.

In the determination of the background and observation error covariances, another problem becomes clear. There are on average only 90 observation stations operating for ozone in the EMEP network in a typical hour. However, it is not the same 90 stations are operating all the time - the location of the measured data is changing. On average the distribution of the stations is not centered around a specific area. If the stations were mainly located around a specific area this could mean that the interpolation operator $\mathbf{H}$ would be sparse with only a few numbers different from zero grouped together. This would give problems when we want to create $\mathbf{HBH}^T$ because the background error covariances matrix should be positive definite. This could result in a singular matrix and the data assimilation analysis would not be feasible. In order to avoid this problem we decide to let all the observation stations go in to $\mathbf{H}$ and let missing measurements be controlled by the departures $\boldsymbol{d} = \boldsymbol{y}_o - \mathbf{H}_{\boldsymbol{x}_a}$. The value zero is assigned to the missing measurements. In the final construction when $\mathbf{d}$ is multiplied by the Kalman gain matrix $\mathbf{K}$, the zero value from the missing measurement would cancel the contribution to $\boldsymbol{x}_a$.

For estimating the background error covariance using the Hollingsworth method, a period of 6 months (April-September 1999) was used as a study period, and both measurements and model results were available for ozone. The

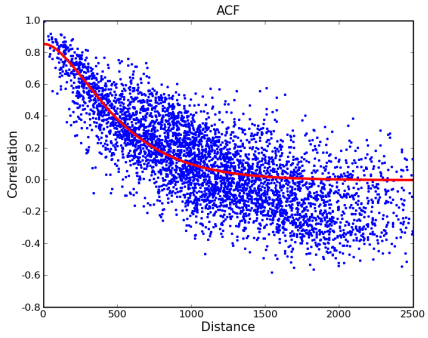


Fig. 1. The correlation function (3) is fitted to the departures correlation as a function of the distance between them in km.

departures at each observation station were calculated at 4 pm every day when the air pollution was well mixed. Furthermore, the maximum values of ozone are typically observed during the afternoon. From this departure the correlation with all the other departures was plotted as a function of their separation. The results can be seen in Figure (1).

From Figure (1) we want to fit the correlation function (3) with the data obtained from the six-months correlation study. In Figure (1) the curve represents the correlation function. Now we are able to determine the background error covariances σ_b^2 and the observation error covariances σ_o^2. The background error covariance can be determined from the interception with y-axes. From the interception we get the correlation $\varrho_b = \frac{\sigma_b^2}{\sigma_b^2+\sigma_o^2}$ and the observation error covariance can then be determined from the simple relation $\varrho_o = 1 - \varrho_b$. The final parameter that can be determined from the auto correlation function is the correlation length, L. As already discussed in the previous section, the correlation length is the distance at which two independent observation stations can be correlated in the model. Beyond that distance the stations will not be correlated in the model. In this study, we found the following parameters for surface ozone $\varrho_b = 0.86$, $\varrho_o = 0.14$ and L = 270 km.

The estimated covariances from the analysis will vary over the seasons and over the local regions i.e. Southern and Northern Europe. The ideal correlation function should be adapted to the fit the local regions and be varied over different seasons. However, in this implementation the basic correlation function will only be tested to determind the effects of the error covariance in the data assimilation routine. Experiments with finding proper correlation functions have been carried out by (Houtekamer et al., 1998), (Hamill et al., 2001) for the EnKF. Finding better error covariances is a investigation in itself and is beyond the scope of this paper.

5 The data assimilation experiments

In these experiments, the data assimilation algorithm is implemented into the DEOM and the effects of applying the algorithm with different configurations are tested against measurements. First, the DEOM was run for the test period from April to September, 1999, to make a reference analysis. The summer period is chosen because there are more ozone episodes in the summer months, which is mainly due to warmer temperatures and much higher global radiation in these periods compared to winter periods. In the following tests the model results are compared to measurements and the improvements relative to the reference run without the data assimilation are examined. Improvements in both the correlation and bias should be expected, since the discrepancies between the observations and the model results have been used to adjust the model results with a weight function. The test period was chosen because it was a well documented period with several ozone episodes and a relatively large temporal spatial coverage of the measurements from the EMEP network.

In this study the tests will be concentrated on the daily maximum values of ozone concentrations. The DEOM model usually performs well with respect to predicting the daily maximum values, which means that the background field from the DEOM model will be less erroneous, compared to other parameters. In this study, it is believed that the data assimilation will decrease the bias and increase the correlation and hence decrease the normalized mean square error, when compared to the measurements.

The measurement data from the EMEP ozone network includes 207 observation stations within the DEOM model grid. All the tests will be conducted over the entire period of 6 months. The data assimilation routine is activated once every day at 12 UTC, unless otherwise stated in the description of the tests. The analyzed model fields are compared to the same observation stations that are used in the data assimilation analysis, but at a different time. The comparison is made for the daily maximum ozone concentration, which usually takes place 4-6 hours (at 16 UTC - 18 UTC) later than when the assimilation procedure was conducted. This gives a separation in time between the assimilation time and the actual comparison time of 4-6 hours.

Another way of evaluating the assimilation process could be to use only half of the observation stations in the data assimilation and use the other half as control/validation stations. This approach should give some information about the spatial separation that arises from the missing observation stations and the stations that are included in the analysis. When the analysis is compared to the observation stations that were excluded in the analysis, the improvement in the analysis field should be seen. However, the number of measurement stations is relatively small, and as mentioned above, the time separation between the observations used for assimilation and the observations used for validation for the daily ozone maximum should be large enough to avoid problems, since the ozone concentrations are transported and chemically produced in the model domain between the time of assimilation and time of validation.

Nine different model runs were performed with the data assimilation algorithm implemented in the DEOM, to carry out the eight experiments, besides a reference run. The model runs are:

1. Reference: The reference run of the DEOM model without the data assimilation routine activated.
2. Experiment 1: The assimilation algorithm conducted with correlation function (3) using equal weights i.e. σ_b^2 =1 , σ_o^2 = 1 and L = 3 grid units (in this case corresponding to 150 km)
3. Experiment 2: Run with optimal weights found by the Hollingsworth method
4. Experiment 3: As experiment 2 with the assimilation routine activated three times a day, on 10 UTC, 11 UTC and 12 UTC.
5. Experiment 4: Run with the anisotropic correlation function (8) with determined weights.
6. Experiment 5: As experiment 2 with the correlation function taking into account the density of observations by (4).
7. Experiment 6: Combination of experiments 4 and 5 with both the anisotropic and the density of observations correlation function. The assimilation routine is activated once per day at 12 UTC.
8. Experiment 7: As experiment 6 with the assimilation routine activated three times a day, at 10 UTC, 11 UTC and 12 UTC.
9. Experiment 8: Run with the correlation function with optimal weights as in experiment 2, adjusted with the formula as in experiment 5 and with the assimilation routine activated three times a day, on 10 UTC, 11 UTC and 12 UTC as in experiment 3.

For all the experiments described above, the model results of the daily maximum value of ozone was validated against measurements from EMEP and examined in the following three different ways (corresponding to average over space, no averaging and average over time, respectively):

1. Time series of the daily maximum value as mean over all stations, where all the observations and calculated values are averaged over space for every day and plotted as function of time.
2. Scatter plots of the daily maximum value including the observations and calculated daily maximum values for all times and locations.

3. Scatter plots of the mean of the daily maximum value at each station, where the observations and the calculated daily maximum values for all stations are averaged over the time period.

5.1 Statistical results from the experiments

In the following subsections, the DEOM model results from all the experiments combined with the different ways of averaging compared to measurements are given. The model results were compared to measurements, and statistics were calculated for every experiment. The statistics are the correlation coefficient, the student's t-test for significance of the correlation coefficient, the fractional bias, and the normalized mean square error.

The statistics for the whole period April-September 1999 for the daily maximum values of ozone from all experiments 1-9 and for the different averaging methods described above are presented in tables 1, 2, and 3.

The great number of statistics from the different assimilation scenarios made a direct comparison difficult. Therefore a ranking system was used to determine the best performing configuration of the data assimilation setup. In the ranking system ranks were assigned as the number 1 for the experiment with the best statistic, 2 for the second best, and so on up to 8. If two statistics had the same value they were assigned the same rank, and the successive rank was skipped. Only the corresponding statistics were compared with each other. In the end the best performing assimilation setup could be determined from the ranking with the lowest total value. Results from the ranking can be seen in table 4. The ranking was performed for each month, April to September, and one ranking for the entire period.

From table 4 it is clear that the assimilation experiment 8 is the best performing including the combination of determining the covariances using the Hollingsworth method, varying the correlation length according to the number of adjacent observation stations and applying the assimilation routine at three successive hours. It can been seen that the correlation coefficient is improved by 0.21 and the students *t* test has gone up by 50.7. The fractional bias and normalized mean square error have decreased by 1.8×10^{-3} and 1.7×10^{-2}, respectively. Having a variable correlation length increases the correlation for stations that are adjacent. It can be seen from statistics from individual stations (not shown here) that the performance improved for these kind of stations.

In all the experiments where hourly successive assimilation was conducted, the model performance is improved. This is clear because more information from the observations is used to correct the background field. This suggests that doing sequential assimilation like from the Ensemble Kalman filter or 4D variational assimilation would enhance the model performance significantly by updating the model at every observations time.

From the ranking table it can be seen that the decomposition of the correlation length into two lengths determined from the wind directions performed worst of all scenarios. This could be due to the way we determined the size of the correlation ellipse, where the size of the perpendicular correlation length was determined from the wind speed ratio v/u. The wind ratio could make the ellipse too narrow so that observation spreading could be too small in some areas. Also experiments 6 and 7 did not perform well, which can be explained from the results from the anisotropic error covariance matrix, which destroys the signal from the observations stations to the model in these experiments too. It should be noted that experiment 3 with the determined error covariances performs much better than the experiment 2 with equal weights. Determining the weights is the most logical way to bring information from the model error and the observation error into the assimilation routine. As stated earlier the covariances was determined from a long time period, which might not be optimal for all time periods, where the weights are less representative.

5.2 Direct comparison of the reference model run and the best performing configuration

In this subsection the visualization results from the reference model run and the best performing model results from experiment 8 are shown.

In Figure 2 the time series of the observations and the model calculations as mean over all stations from the EMEP network are shown. The figure includes times series of daily mean, hourly values and daily maximum values. From the daily mean and the daily maximum values it becomes clear that the assimilation routine pulls the model calculations toward the measurements and thereby decreases the fractional bias and increases the correlation between the observed and modeled time series.

In Figures 3 and 4 the frequency distributions of the three statistics, calculated at the individual measurement stations for the period April-September 1999, are compared to the reference for the daily mean and daily maximum values, respectively. The figures show that the assimilation routine significantly increases the correlation for a number of stations, which can seen in the way the histogram shifts to the right compared to the reference. The fractional bias and the normalized mean square error are relatively small in both figures for most of the measurement stations. A small change in the fractional bias, which has a tendency to be centered more around zero, can be observed from the figures. Furthermore, a shift towards smaller values of the $NMSE$ is seen.

In the scatter plot in Figure 5 showing the daily mean ozone values, we can see that the assimilation routine again improves the model outcome in the way the scatter plot gets more trimmed around the 1:1 line and the correlation coefficient increases from 0.49 to 0.68. The same is true for the scatter plots shown in Figure 6 including all the daily maxi-

Model run	Correlation coefficient	Students t test	Fractional bias	NMSE
Reference	0.86	22.9	5.2E-03	3.1E-03
Experiment 1	0.94	36.6	3.8E-03	1.4E-03
Experiment 2	0.94	38.6	1.8E-03	1.3E-03
Experiment 3	0.96	47.6	8.4E-03	1.0E-03
Experiment 4	0.93	34.7	9.8E-03	1.6E-03
Experiment 5	0.95	39.2	5.1E-03	1.3E-03
Experiment 6	0.94	36.7	7.3E-03	1.5E-03
Experiment 7	0.95	41.8	1.1E-02	1.2E-03
Experiment 8	0.96	45.1	3.4E-03	1.0E-03

Table 1. *Statistics calculated for the reference model run and experiments 1-8 compared to measured data from EMEP, covering the period April-September 1999, based on the time series of the daily maximum value as mean over all stations, where all the observations and calculated values are averaged over space for every day and plotted as function of time.*

Model run	Correlation coefficient	Students t test	Fractional bias	NMSE
Reference	0.62	101.5	5.2E-03	4.5E-02
Experiment 1	0.72	131.4	3.8E-03	3.3E-02
Experiment 2	0.73	137.8	1.8E-03	3.2E-02
Experiment 3	0.76	149.8	8.5E-03	2.9E-02
Experiment 4	0.74	143.1	5.1E-03	3.0E-02
Experiment 5	0.71	129.8	9.7E-03	3.4E-02
Experiment 6	0.73	138.1	7.2E-03	3.1E-02
Experiment 7	0.75	143.7	1.1E-02	3.0E-02
Experiment 8	0.76	152.2	3.4E-03	2.8E-02

Table 2. *Statistics calculated for the reference model run and experiments 1-8 compared to measured data from EMEP, covering the period April-September 1999, based on the Scatter plots of the daily maximum value including the observations and calculated daily maximum values for all times and locations.*

Model run	Correlation coefficient	Students t test	Fractional bias	NMSE
Reference	0.67	8.7	4.6E-03	9.1E-03
Experiment 1	0.74	10.6	4.4E-03	7.6E-03
Experiment 2	0.76	11.1	2.4E-03	7.3E-03
Experiment 3	0.78	12.0	9.0E-03	7.0E-03
Experiment 4	0.75	10.9	9.0E-03	7.5E-03
Experiment 5	0.78	11.9	4.5E-03	6.7E-03
Experiment 6	0.77	11.6	6.6E-03	7.0E-03
Experiment 7	0.80	12.8	1.1E-02	6.4E-03
Experiment 8	0.81	13.2	3.0E-03	3.0E-03

Table 3. *Statistics calculated for the reference model run and experiments 1-8 compared to measured data from EMEP, covering the period April-September 1999, based on the Scatter plots of the mean of the daily maximum value at each station, where the observations and the calculated daily maximum values for all stations are averaged in time.*

Model run	Apr.	May	Jun.	Jul.	Aug.	Sep.	Apr.-Sep.	Total	Global rank
Reference	55	44	69	59	67	62	75	431	7
Experiment 1	53	48	29	52	33	28	50	293	5
Experiment 2	67	63	44	59	49	35	35	352	6
Experiment 3	30	40	27	41	37	9	36	220	2
Experiment 4	90	90	90	85	67	90	72	584	9
Experiment 5	38	28	45	28	29	36	32	236	3
Experiment 6	79	73	81	81	70	78	52	514	8
Experiment 7	29	24	38	22	49	38	46	246	4
Experiment 8	9	10	17	11	17	21	12	97	1

Table 4. *The outcome from the ranking process of the combined ranks of correlation coefficient, fractional bias and the normalized mean square error including the results for daily maximum values averaged in time, not averaged and averaged in space for every month during the period April-September 1999 as well as for the whole period (all). The global rank with the value 1 is the best performing configuration of the model and the value 9 refers to the worst performing model configuration.*

mum values from all the measurement stations as well as the corresponding model results.

In Figures 7 and 8 scatter plots are given, including mean values for all measurement stations for the daily mean and daily maximum values, respectively. In these figures average values are made over time, whereas in Figure 2, the averaging is carried over space. In Figure 7, the correlation coefficient increases from 0.37 to 0.58 and the bias decreases a little. For the daily maximum values displayed in Figure 8, an increase in the correlation coefficient from 0.67 to 0.81 is seen. Also here the bias and the normalized mean square error decrease, as expected. In both figures the increase in the correlation coefficient is significant, which can be seen in the increase of the student's t-test parameter. An increase in the t-test parameter of more than 2.632 means that the increase is significant within a significance level of 1%.

5.3 Analysis of two ozone episodes

In this section, two ozone episodes that occurred on September 7th, 1999 and September 12th, 1999 will be examined. The effect of using the data assimilation algorithm is compared to the reference run where no data assimilation is applied. The model configuration described in experiment 8 is used. The results are presented in Figures 9 and 10, respectively, where the reference run is shown in top figures and the analyzed fields in the lower figures. Both model runs are carried out continuously, starting on September 1st, with initial data from a previous run for the month before. In the model run using the data assimilation technique, the data is assimilated each day at 10 UTC, 11 UTC and 12 UTC.

For both episodes there are some differences between the reference and the analyzed fields. This is the case especially for September 7th, see Figure 9, where the ozone concentrations in the Mediterranean area are decreased considerable. In this area the assimilation algorithm has pulled the general concentration level down. Also in central Europe and in the Scandinavia region, ozone concentrations are lower compared to the reference. For September 12th, see Figure 10, the differences between the reference and the assimilated results are smaller, however, corrections are seen for smaller areas, especially in the area east of Spain and south west of Denmark.

In general, we see that the DEOM model overestimated the ozone concentration for these two days in September 1999. The overall ozone concentrations are corrected towards the observations and thereby improve the prediction capability of the DEOM model.

6 Conclusions

This study reports the first results of a data assimilation routine that has been developed based on Statistical Interpolation for the DEOM model. Eight different experiments including different configurations of the data assimilation algorithm were defined and tested against measurements from the EMEP network for the period April-September 1999. In order to find the optimal configuration of the algorithm, the model results from the different experiments were ranked according to the performance.

The Statistical Interpolation algorithm significantly improved the performance of the DEOM model when compared to the measurements. The Statistical Interpolation algorithm generally improved the correlation coefficient with 0.10 and the fractional bias with 2×10^{-3} and normalized mean square error with 2×10^{-2} for the overall ozone daily maximum concentrations.

The best performing setup of the data assimilation algorithm was found to be the configuration in experiment 8, including the combination of determining the covariances using the Hollingsworth method, varying the correlation length according to the number of adjacent observation stations and applying the assimilation routine at three successive hours during the morning at 10 UTC, 11 UTC and 12 UTC. The results from the experiments have shown that the data assimilation routine together with a CTM is beneficial for obtaining better performance of the short-term ozone forecasts

using the CTM model. Improvements in the correlation coefficients in the range of 0.1 to 0.21 between the reference and configuration in experiment 8 were seen. Additionally, there were significant reductions in bias and NMSE.

Two ozone episodes that occurred on September 7th, 1999 and September 12th, 1999 were examined in order to make visual testing of the behavior of the algorithm for artificial behavior. It was concluded from this experiment that the data assimilation routine did not introduced any sharp gradients into the model which could lead to artificial model solutions.

It was expected that the data assimilation routine should have some effect on e.g. the NO_2 concentrations when altering the ozone concentrations. In experiment 8 there was no clear indication that the NO_2 concentration was effected significantly (not shown here). In the next step the NO_2 measurements could also be assimilated into the DEOM model. However, the measurement of NO_2 is only given as daily mean values. This means that the measurements cannot be used directly as was the case for the ozone measurements, where the hourly values were more representative for the model time step. Methods for correct representation of the daily measurements in the DEOM model using data assimilation can probably be developed by e.g. assimilating daily fields into daily mean values from the model, and then using the fraction between the two to adjust the NO_2 concentration at higher time resolution. This requires, however, a number of new tests and is beyond the scope of this paper. A next step of using the algorithm will be operational data assimilation of NO_2 data from satellite measurements.

Acknowledgements. This paper was partly supported by the project "GMES Service Extensions for Denmark: EO Improved Air Quality Forecast", funded by the European Space Agency under the umbrella of PROMOTE, and partly by the Danish Research School ITMAN.

References

Balgovind, R., Dalcher, A., Ghil, M., and Kalnay, E.: A Stochastic-Dynamic Model for the Spatial Structure of Forecast Error Statistics, Monthly Weather Review, 111, 701–722, 1983.

Blond, N. and Vautard, R.: Composition and Chemistry - D17303 - Three-dimensional ozone analyses and their use for short-term ozone forecasts (DOI 10.1029/2004JD004515), Journal of Geophysical Research - Part D - Atmospheres, 109, 2004.

Blond, N., L.Bel, and R.Vautard: Three-dimensional ozone data analysis with an air quality model over the Paris area, Journal of Geophysical Research, 108, 4744, 2003.

Bouttier, F. and Courtier., P.: Data assimilation concepts and methods., Tech. rep., ECMWF training course lecture notes, 1998.

Brandt, J., Christensen, J., Frohn, L., Berkowicz, R., and Palmgren, F.: The DMU-ATMI THOR Air Pollution Forecast System. System Description, 2000.

Brandt, J., Christensen, J. H., Frohn, L. M., and Berkowicz, R.: Operational air pollution forecasts from regional scale to urban street scale. Part 1: system description, Physics and Chemistry of the Earth,Part B: Hydrology,Oceans and Atmosphere, 26, 781–786, 2001a.

Brandt, J., Christensen, J. H., Frohn, L. M., and Berkowicz, R.: Operational air pollution forecasts from regional scale to urban street scale. Part 2: performance evaluation, Physics and Chemistry of the Earth,Part B: Hydrology,Oceans and Atmosphere, 26, 825–830, 2001b.

Brandt, J., Christensen, J. H., Frohn, L. M., Palmgren, F., Berkowicz, R., and Zlatev, Z.: Operational air pollution forecasts from European to local scale, Atmospheric Environment, 35, 91–98, 2001c.

Brandt, J., Christensen, J., Frohn, L. M., Berkowicz, R., Skjøth, C. A., Geels, C., Hansen, K. M., Frydendall, J., Hedegaard, G. B., Hertel, O., Jensen, S. S., Hvidberg, M., Ketzel, M., Olesen, H. R., Løfstrøm, P., and Zlatev, Z.: THOR - an operational and integrated model system for air pollution forecasting and management from global to local scale, in: Proceedings from the First ACCENT Symposium, Urbino, Italy, 12th-16th September 2005, p. pp. 6, ACCENT, 2005.

Burges, G., van Leeuwen, P. J., and Evensen, G.: Analysis Scheme in the Ensemble Kalman Filter, American Meteorological Society, 126, 1719–1724, 1998.

Daley, R.: Atmospheric Data Analysis, Cambridge University Press, 1996.

Denby, B., Kahnert, M., Brandt, J., Frydendall, J., and Zlatev, Z.: Development and application of data assimilation in regional scale atmospheric chemistry models, Tech. Rep. NILU O 30/2007, Ref: O-105077, April 2008, pp. 47, 2008.

Elbern, H. and Schmidt, H.: A four-dimensional variational chemistry data assimilation scheme for Eulerian chemistry transport modeling, Journal of Geophysical Research-Atmospheres, 104, 18 583–18 598, 1999.

Elbern, H., Schmidt, H., and Ebel, A.: Variational data assimilation for tropospheric chemistry modeling, Journal of Geophysical Research-Atmospheres, 102, 15 967–15 985, 1997.

Elbern, H., Schmidt, H., Talagrand, O., and Ebel, A.: 4D-variational data assimilation with an adjoint air quality model for emission analysis, Environmental Modelling & Software, 15, 539–548, 2000.

Evensen, G.: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, Journal of Geophysical Research, 99, 10 143–10 162, 1994.

Gandin, L. S.: Objective Analysis of Meteorological Fields, in: Program Scientific Translalions, 1963.

Gaspari, G. and Cohn, S. E.: Construction of correlation functions in two and three dimensions, Quarterly Journal of the Royal Meteorological Society, 125, 723–757, 1999.

Hamill, T. M., Jeffrey, Whitaker, S., and Snyder, C.: Distance-Dependent Filtering of Background Error Covariance Estimates in an Ensemble Kalman Filter, 2776 MONTHLY WEATHER REVIEW VOLUME 129 Monthly Weather Review, 129, 27762790, 2001.

Hertel, O., Ellermann, T., Palmgren, F., Berkowicz, R., Løfstrøm, P., Frohn, L. M., Geels, C., Skjøth, C. A., Brandt, J., Christensen, J., Kemp, K., and Ketzel, M.: Integrated Air Quality Monitoring - Combined use of measurements and models in monitoring programmes, Environmental Chemistry, 4, 65–74, 2007.

Hesstvedt, E., Hov, O., and Isaksen, I. S. A.: Quasi-Steady-State

Approximations in Air-Pollution Modeling - Comparison of Two Numerical Schemes for Oxidant Prediction, International Journal of Chemical Kinetics, 10, 971–994, 1978.

Hoelzemann, J. J., Elbern, H., and Ebel, A.: PSAS and 4D-var data assimilation for chemical state analysis by urban and rural observation sites, Physics and Chemistry of the Earth Part B-Hydrology Oceans and Atmosphere, 26, 807–812, 2001.

Hollingsworth, A. and Lonnberg, P.: The Statistical Structure of Short-Range Forecast Errors As Determined from Radiosonde Data .1. the Wind-Field, Tellus Series A-Dynamic Meteorology and Oceanography, 38, 111–136, 1986.

Houtekamer, P. L., Mitchell, and Herschel, L.: Data Assimilation Using an Ensemble Kalman Filter Technique, Monthly Weather Review, 126, 796811, 1998.

Ide, K., Courtier, P., Ghil, M., and Lorenc, A. C.: unified Notation for Data Assimilation: operational, Sequential and Variational, J. Met. Soc. Japan, Special Issue on Data Assimilation in Meteorology and Oceanography: Theory and Practice, 181–189, 1997.

Kang, D., Mathur, R., Rao, T. S., and Yu, S.: Bias adjustment techniques for improving ozone air quality forecasts, Journal of Geophysical Research - Atmospheres, 113, D23 308+, doi:10.1029/2008JD010151, http://dx.doi.org/10.1029/2008JD010151, 2008.

Lamarque, J. F., Khattatov, B. V., Gille, J. C., and Brasseur, G. P.: Assimilation of Measurement of Air Pollution from Space (MAPS) CO in a global three-dimensional model, Journal of Geophysical Research-Atmospheres, 104, 26 209–26 218, 1999.

Lawrence, M. G., Hov, ., Beekmann, M., Brandt, J., Elbern, H., Eskes, H., Feichter, H., and Takigawa, M.: The Chemical Weather, Environmental Chemistry, 2, 6–8, 2005.

Lonnberg, P. and Hollingsworth, A.: The Statistical Structure of Short-Range Forecast Errors As Determined from Radiosonde Data .2. the Covariance of Height and Wind Errors, Tellus Series A-Dynamic Meteorology and Oceanography, 38, 137–161, 1986.

McRae, G. J., Goodin, W. R., and Seinfeld, J. H.: Numerical-Solution of the Atmospheric Diffusion Equation for Chemically Reacting Flows, Journal of Computational Physics, 45, 1–42, 1982.

Ménard, R. and Chang, L. P.: Assimilation of Stratospheric Chemical Tracer Observations Using a Kalman Filter. Part II: chi2-Validated Results and Analysis of Variance and Correlation Dynamics, Monthly Weather Review, 128, 2672–2686, 2000.

Ménard, R., Cohn, S. E., Chang, L. P., and Lyster, P. M.: Assimilation of Stratospheric Chemical Tracer Observations Using a Kalman Filter. Part I: Formulation, Monthly Weather Review, 128, 2654–2671, 2000.

Tilmes, S., Brandt, J., Flatoy, F., Bergstrom, R., Flemming, J., Langer, J., Christensen, J. H., Frohn, L. M., Hov, O., Jacobsen, I., Reimer, E., Stren, R., and Zimmermann, J.: Comparison of five eulerian air pollution forecasting systems for the summer of 1999 using the german ozone monitoring data, Journal of Atmospheric Chemistry, 42, 91–121, 2002.

van Loon, M. and Heemink, A. W.: Kalman filtering for nonlinear atmospheric chemistry models: first experiences, Tech. Rep. MAS-R9711, Centrum voor Wiskunde en Informatica, 1997.

Zlatev, Z. and Brandt, J.: Testing Variational Data Assimilation Modules, Springer-Verlag, pp.395-402, 2005.

Zlatev, Z. and Brandt, J.: Testing the accuracy of a data assimilation algorithm, International Journal of Computational Science and Engineering, 3, 305–313, 2007.

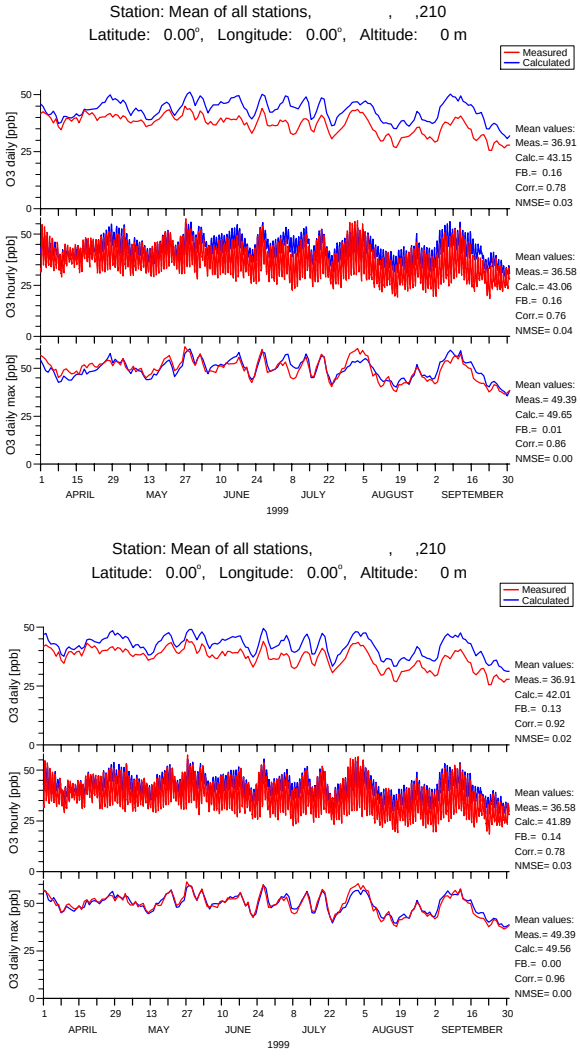


Fig. 2. *Time series, covering the period April-September 1999, of measured and modelled values taken as a mean over all the measurement stations in the EMEP network of daily mean, hourly and daily maximum values of O_3. The upper figure shows the results from the reference run without applying the data assimilation technique. The lower figure shows the results from a model run including the data assimilation as the configuration in experiment 8.*

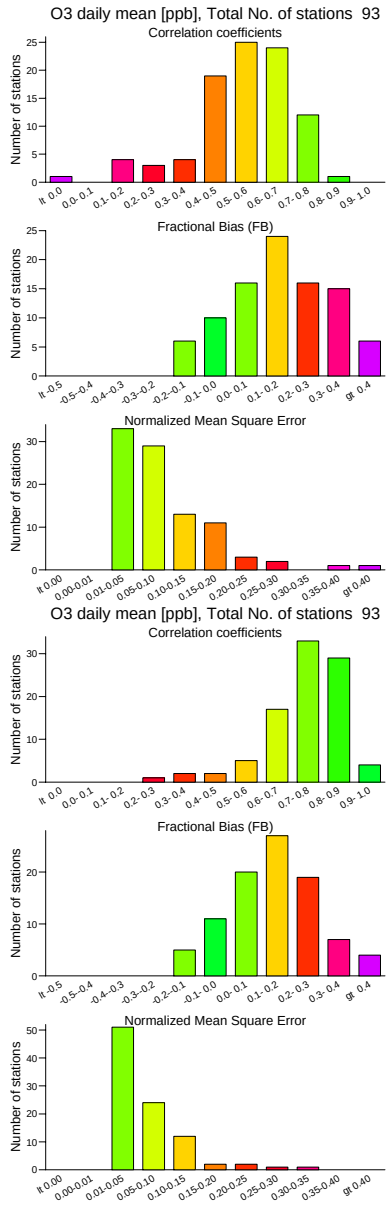


Fig. 3. *Frequency distributions for the correlation coefficient, the fractional bias and the normalized mean square error, including the statistics from comparisons between measurements and model results for the daily mean values at each measurement stations within the EMEP network. The testing period is April-September 1999. The upper figure shows the results from the reference run without applying the data assimilation technique. The lower figure shows the results from a model run including the data assimilation as the configuration in experiment 8.*

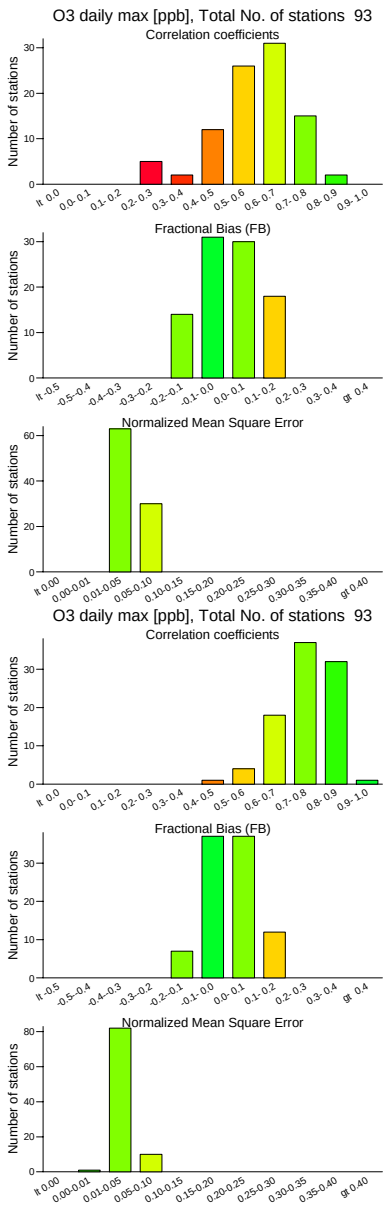


Fig. 4. *Frequency distributions for the correlation coefficient, the fractional bias and the normalized mean square error, including the statistics from comparisons between measurements and model results for the daily maximum values at each measurement stations within the EMEP network. The testing period is April-September 1999. The upper figure shows the results from the reference run without applying the data assimilation technique. The lower figure shows the results from a model run including the data assimilation as the configuration in experiment 8.*

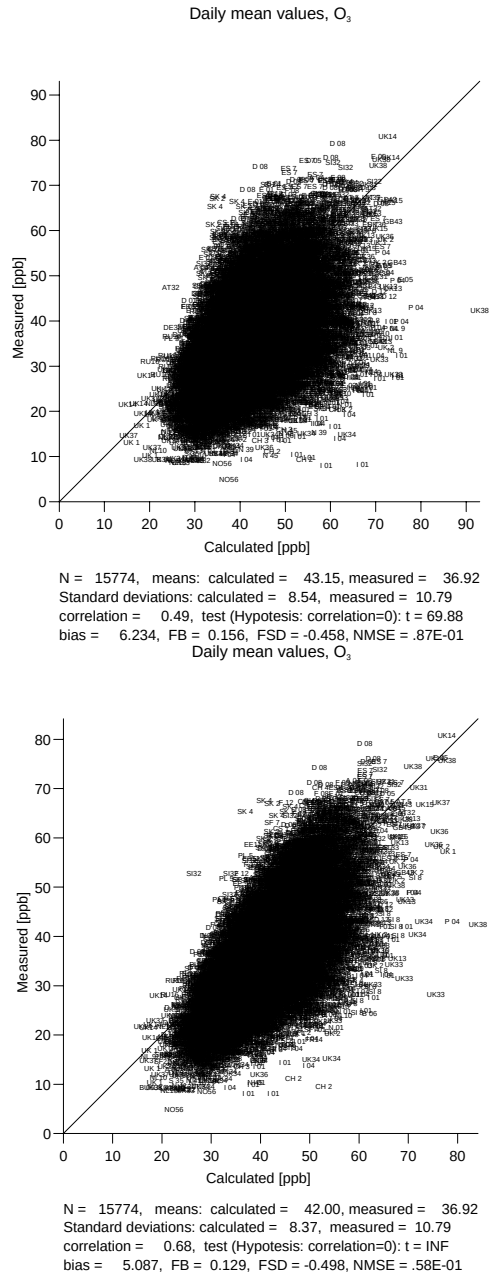


Fig. 5. *Scatter plot showing a comparison between modelled and measured values of the daily mean values of O_3 including all values during the period April-September 1999, at all stations in the EMEP network. The upper figure shows the results from the reference run without applying the data assimilation technique. The lower figure shows the results from a model run including the data assimilation as the configuration in experiment 8.*

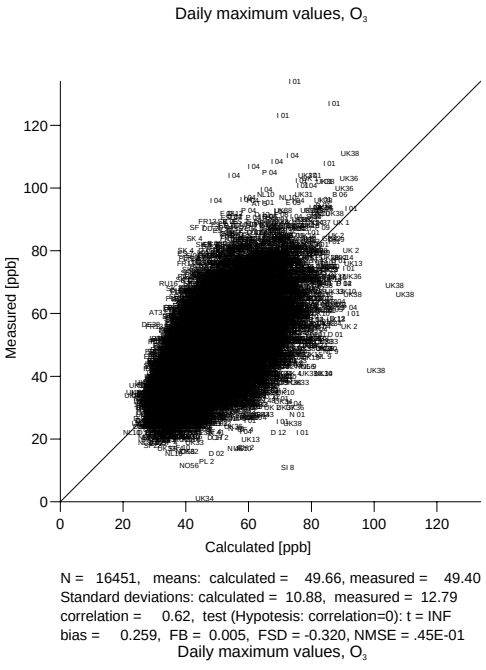


Fig. 6. *Scatter plot showing a comparison between modelled and measured values of the daily maximum values of O_3 including all values during the period April-September 1999, at all stations in the EMEP network. The upper figure shows the results from the reference run without applying the data assimilation technique. The lower figure shows the results from a model run including the data assimilation as the configuration in experiment 8.*

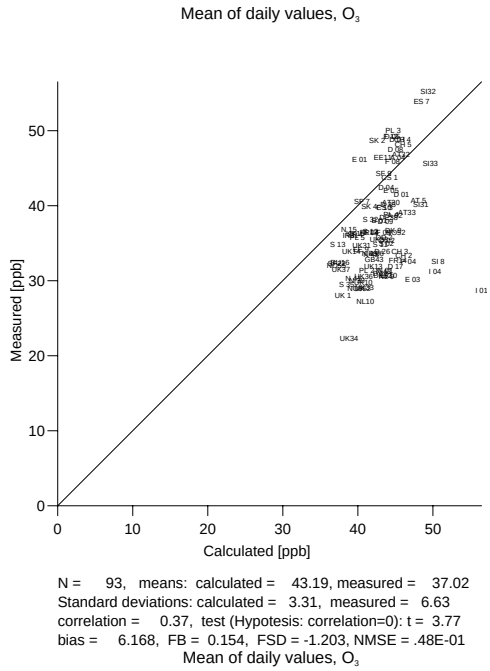


Fig. 7. *Scatter plot showing a comparison between modelled and measured values of the daily mean values of O_3 taken as a mean over the period April-September 1999, at all stations in the EMEP network. The upper figure shows the results from the reference run without applying the data assimilation technique. The lower figure shows the results from a model run including the data assimilation as the configuration in experiment 8.*

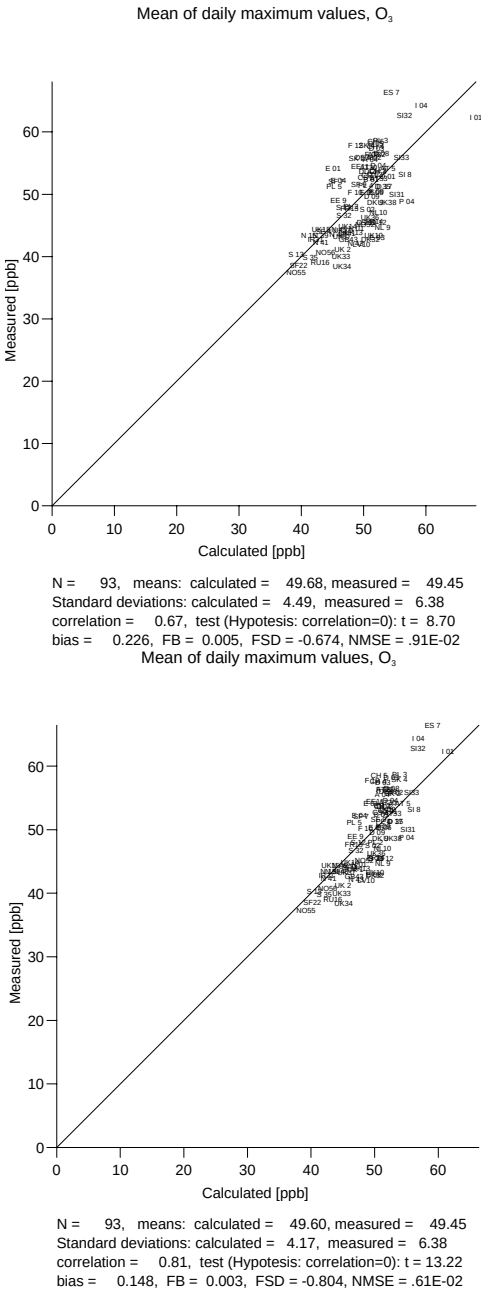


Fig. 8. *Scatter plot showing a comparison between modelled and measured values of the daily maximum values of O_3 taken as a mean over the period April-September 1999, at all stations in the EMEP network. The upper figure shows the results from the reference run without applying the data assimilation technique. The lower figure shows the results from a model run including the data assimilation as the configuration in experiment 8.*

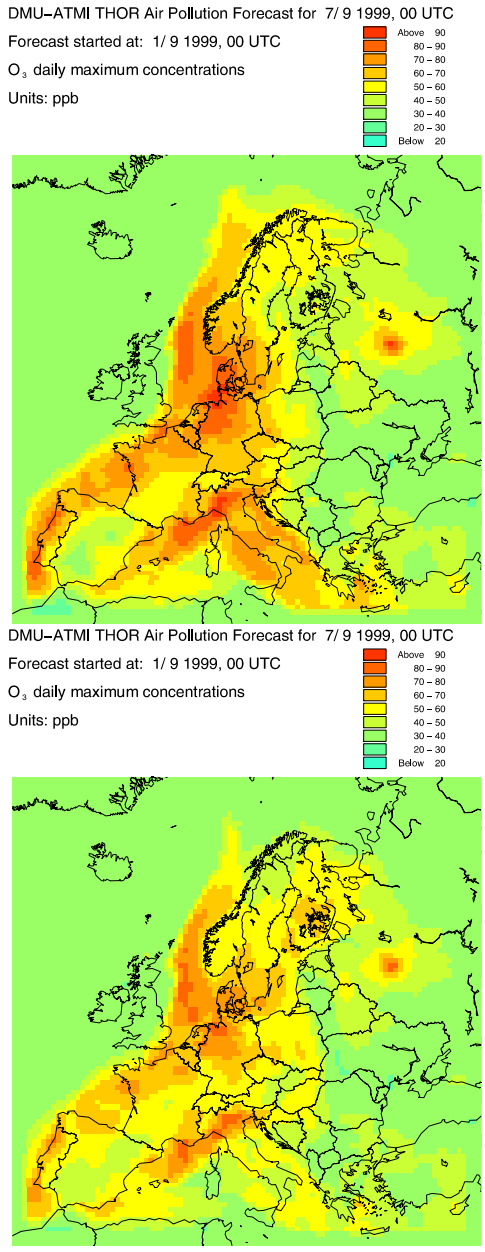


Fig. 9. *Daily maximum ozone concentrations calculated using DEOM during an ozone episode in Europe in September 7, 1999. The upper figure shows the results from the reference run without using data assimilation. The lower figure shows the corresponding result including the data assimilation of surface O_3 based on the configuration in experiment 8.*

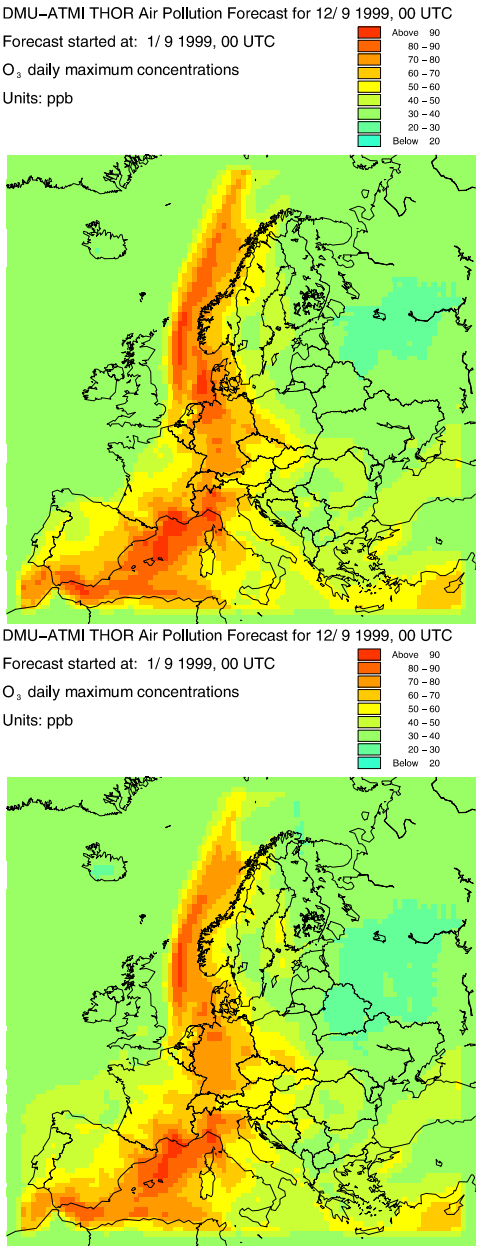


Fig. 10. *Daily maximum ozone concentrations calculated using DEOM during an ozone episode in Europe in September 12, 1999. The upper figure shows the results from the reference run without using data assimilation. The lower figure shows the corresponding result including the data assimilation of surface O_3 based on the configuration in experiment 8.*

APPENDIX D

Paper D

Accepted as conference proceed to 7th International Conference on Hydroinformatics HIC 2006, Nice, France

7th International Conference on Hydroinformatics
HIC 2006, Nice, FRANCE

SURROGATE MODELLING IN MARINE MODELLING: UNCERTAINTY PROPAGATION BY REDUCED MODELS

JACOB V. TORNFELDT SØRENSEN
DHI Water & Environment, Agern Allé 5
Hørsholm, DK-2970, Denmark

JAN FRYDENDALL
DHI Water & Environment, Agern Allé 5
Hørsholm, DK-2970, Denmark

HENRIK MADSEN
DHI Water & Environment, Agern Allé 5
Hørsholm, DK-2970, Denmark

Surrogate modeling is a methodology in which complex models are replaced by simpler models for certain applications that benefit from trading a degree of accuracy for computational speed. One particular aspect of surrogate modeling is investigated in this paper for the purpose of performing Monte Carlo uncertainty simulations at high speed. The complex model considered is the finite difference barotropic model, MIKE 21, set up for the artificial Ideal Bay. The surrogate modeling technique applied is based on a first order Taylor Series expansion of this barotrotic model in a reduced space spanned by covariance eigenvectors derived from an empirical orthogonal function analysis. The study demonstrates that a significantly smaller computational effort can provide uncertainty estimates that resemble Monte Carlo estimates using the high-dimensional complex model. This result is promising and the methodology may be generalized into operationally efficient techniques in data assimilation and parameter estimation through both sequential and adjoint approaches.

INTRODUCTION

Numerical modelling techniques are consolidated, flexible and widely applicable for solving a range of problems in the marine and coastal environment. Despite increasing computational power and the resulting increase in detail and areal coverage, these techniques intrinsically suffers from over parameterisation in a given application and uncertain parameters, initial and boundary conditions.

Surrogate modelling is introduced here as a mean to approximate the numerical modelling engines in a reduced space optimised to capture the propagation of model uncertainty in a particular set-up. Many basic features follows [1]. Basically the approach requires a set of states from the high-dimensional model that spans the dynamical range

to be captured by the surrogate model. An empirical orthogonal function (EOF) analysis is performed on this set to obtain an optimal set of orthogonal vectors that spans the model states. Next a reference model simulation must be established. Here, care must be taken to choose a reference that is itself a solution to the numerical model. Thereafter a surrogate model can be constructed by using the numerical model to calculate the reduced operator from one or more reduced directions in state space to all other directions in the reduced space. For a simple 1st order Taylor series expansion this only requires a number of time steps to be performed that corresponds to the desired degrees of freedom in the reduced surrogate model.

For demonstration, a surrogate model will be constructed for performing a Monte Carlo propagation of model uncertainty assuming errors in water level open boundaries and wind velocity components in a simple barotropic model set-up. The accuracy of the approach is assessed by comparing the results to a full Monte Carlo propagation.

METHODOLOGY

The challenge in surrogate modeling is to develop a model of a model. Hence, the starting point will always be a well functioning set-up of a numerical ocean model in an area of interest. This model must then be approximated in a number of suitable ways which are capable of preserving the main modes of variability in a particular application of the high-dimensional model, but much faster. The surrogate modeling framework considered herein, consists of two main aspects: 1) Reduction of the model dimension and 2) approximation of the model dynamics.

The theoretical basis for developing a surrogate model of a marine system thus consists of the following elements:

- High–dimensional, non-linear numerical model
- Low-dimensional state space approximation based on EOF analysis
- Taylor series expansion of the numerical model

High dimensional non linear numerical model

The performance of high-dimensional numerical models is of key importance for any surrogate modeling technique. In the present study the two-dimensional ocean modeling software MIKE 21, is applied in a barotropic set-up in the artificial “Ideal Bay”.

The model can be described as

$$\boldsymbol{x}^f(\boldsymbol{t}_{i+1}) = \boldsymbol{M}\left[\boldsymbol{x}^f(\boldsymbol{t}_i), \boldsymbol{u}_{i+1}, \boldsymbol{\alpha}_{i+1}\right] \quad (1)$$

, where $\boldsymbol{M}$ is the high–dimensional, non-linear model operator and $\boldsymbol{x}^f$ is the model state composed of water levels and fluxes in each grid point in the model domain. The time varying forcing in the model is represented by $\boldsymbol{u}$ and model parameters by $\boldsymbol{\alpha}$. This and similar numerical model engines have been developed to obtain a high flexibility and to include physical phenomena encountered in a wide range of applications. A key

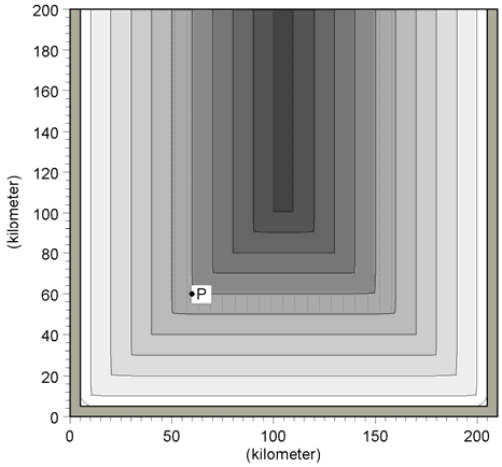


Figure 1 Bathymetri of Ideal Bay. The position P is shown

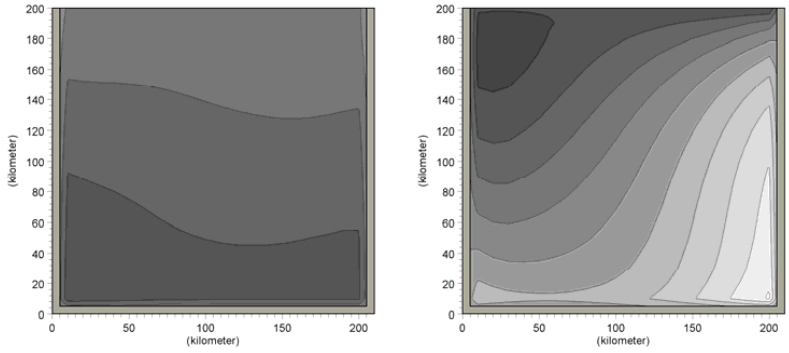
parameter when applying the model in a particular setup, is to have sufficient spatial and temporal resolution to resolve the physical processes considered. This however, introduces a high computational burden and discretisation techniques such as dynamical nesting, curvilinear coordinates and flexible meshes have been developed to allow the computational effort to be focused at the area of interest and the related processes. The surrogate modeling introduced here tries to achieve the same by maintaining only the main part of the spatially coherent dynamical variability and neglecting the rest.

In the present study a dynamically nested barotropic model of the artificial Ideal Bay is used for demonstration of the proposed surrogate modeling approach. The area can be seen in Figure 1. For validation, the model is run for 3 typical days with 12 hourly periodic 1m water level variation at the open as well as wind induced currents from a constantwestern wind of 20 m/s. As an example of the simulation the time series of water level at position P and is shown as the grey curve in Figure 3 and a snapshot is shown in Figure 4..

Low dimensional state space approximation

The state space is reduced by the application of an Empirical Orthogonal Function (EOF) analysis, [2]. This technique is essentially an eigenvalue decomposition of a covariance matrix constructed from a representative set of model results. The N selected time steps (which need not be consecutive) of model solutions are arranged in a matrix X, such that all spatial information at a given time step is contained in a single row and the

time steps of a particular position in a single column. Hence $C=X^TX$ is the covariance matrix for which an eigenvalue decomposition is performed. The resulting eigenvector in



contained in X. As an example the leading eigenvector pattern for water level in the model area is shown in Figure 2.

In the present analysis, 5 EOF patterns was contained for each dynamical variable, giving a reduced space of rank 15, compared to about 1,600 for the hydrodynamic model.

Taylor series expansion

The governing equations from which the high dimensional model is derived are very difficult to express analytically in the reduced low dimensional state space introduced in section 0. However, the high dimensional model itself can be used to derive the representation of the governing equations in the low dimensional state space. This section will explain how, but as a first step the linearized model equations will be derived. The 1st order Taylor series expansion of the non linear model operator can be written as,

$$\bar{\boldsymbol{x}}^{f}(\boldsymbol{t}_{i+1}) = \boldsymbol{M}_{i+1}\left[\boldsymbol{x}'^{f}(\boldsymbol{t}_i), \boldsymbol{u}'_{i+1}, \boldsymbol{\alpha}'_{i+1}\right] + \sum_{j=1}^{N_x} \frac{\partial \boldsymbol{M}_{i+1}\left[\boldsymbol{x}'^{f}(\boldsymbol{t}_i), \boldsymbol{u}'_{i+1}, \boldsymbol{\alpha}'_{i+1}\right]}{\partial \boldsymbol{x}_j}\Delta \boldsymbol{x}_j + \sum_{j=1}^{N_u} \frac{\partial \boldsymbol{M}_{i+1}\left[\boldsymbol{x}'^{f}(\boldsymbol{t}_i), \boldsymbol{u}'_{i+1}, \boldsymbol{\alpha}'_{i+1}\right]}{\partial \boldsymbol{u}_j}\Delta \boldsymbol{u} + \sum_{j=1}^{N_\alpha} \frac{\partial \boldsymbol{M}_{i+1}\left[\boldsymbol{x}'^{f}(\boldsymbol{t}_i), \boldsymbol{u}'_{i+1}, \boldsymbol{\alpha}'_{i+1}\right]}{\partial \boldsymbol{\alpha}_j}\Delta \boldsymbol{\alpha} \quad (2)$$

In Equation (2) N_x is the size of the state space (number of variables times number of grid points), N_u is the size of the forcing space and N_α is the size of the parameter space. For $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}]^T$ and the definition

$$\bar{\boldsymbol{x}}^{f}(\boldsymbol{t}_{i+1}) = \boldsymbol{x}'^{f}(\boldsymbol{t}_{i+1}) + \Delta \boldsymbol{x}(\boldsymbol{t}_{i+1}) \quad (3)$$

Equation *(2)* can be rewritten,

$$\Delta \boldsymbol{x}^{f}(\boldsymbol{t}_{i+1}) = (\nabla_z \boldsymbol{M})(\Delta z(\boldsymbol{t}_i)) \quad (4)$$

Under the assumption that

$$\boldsymbol{x}'^{f}(t_{i+1}) = \boldsymbol{M}_{i+1}\left[\boldsymbol{x}'^{f}(t_i), \boldsymbol{u}'_{i+1}, \boldsymbol{\alpha}'_{i+1}\right] \tag{5}$$

Hence, if the reference solution is a solution by itself, a model can be constructed for the deviations. For an ocean model, the reference solution could e.g. be a steady solution with no motion, the wind zero and the open boundary equal to the steady solution for water level. This would leave the entire ocean variability to be modeled by the surrogate model, but the solution would be exact only for no dynamical forcing. The background could also be the tidal or another dynamical signal, in which case care must be taken to subtract this signal also in the EOF analysis and in the derivation of the coefficients in the low order model.

Obviously, the idea is to avoid expensive model computations and hence for predictive purposes, the reference solution should be steady or periodic, such that it can be calculated once and for all. Introducing the scaling coefficient β_x, β_u and β_α, the coefficients in $\nabla_z \boldsymbol{M}$ is calculated as

$$\nabla_z \boldsymbol{M} = \begin{bmatrix} \dfrac{\boldsymbol{M}(\boldsymbol{x}'^{f} + \boldsymbol{\beta}_x \Delta \boldsymbol{x}, \boldsymbol{u}'_{i+1}, \boldsymbol{\alpha}'_{i+1})}{\boldsymbol{\beta}_x} \\ \dfrac{\boldsymbol{M}(\boldsymbol{x}'^{f}, \boldsymbol{u}'_{i+1} + \boldsymbol{\beta}_u \Delta \boldsymbol{u}, \boldsymbol{\alpha}'_{i+1})}{\boldsymbol{\beta}_u} \\ \dfrac{\boldsymbol{M}(\boldsymbol{x}'^{f}, \boldsymbol{u}'_{i+1}, \boldsymbol{\alpha}'_{i+1} + \boldsymbol{\beta}_\alpha \Delta \boldsymbol{\alpha})}{\boldsymbol{\beta}_\alpha} \end{bmatrix}^{T} \tag{6}$$

Such a linear expansion holds for and set of basic vectors. E.g. one could principally find the tangent linear model by perturbing each variable at each grid point, [3].

However, with further terms in the Taylor series expansion or another orthogonal function expansion, different possibly non-linear models can be derived, but there is a trade-off with number of times the high dimensional model must be executed. The number of model evaluations needed in the expansion depends on the size, n_{ss}, of the state space in which the expansion id performed as well as the expansion selected. For the first order Taylor series explained above, nss model evaluations are needed, while of the order n_{ss}^2 evaluations are needed for a second order Taylor series.

SURROGATE MODEL CONSTRUCTION

In this section the Taylor series expansion is combined with the reduced model space representation provided by an EOF analysis.

The surrogate model for a complex numerical model can be constructed by perturbing an initial field at rest in the direction of the 15 EOFs selected for the case considered in the Ideal Bay. From the perturbed initial fields one time step (900s) was taken and the results at this point projected back onto the EOFs (simply the dot-products). This gives an 15x15 matrix which constitutes the model, $\mathbf{M_R}$, in the reduced space. Further, the response to forcing was similarly calculated by running the model one time step (again from a state at rest) with a set of perturbed normalized forcing fields. For the wind forcing two components were derived from perturbing the x- and y-components of wind

velocity for the Monte Carlo simulations. For the open boundary one component was derived by perturbing the spatially constant boundary condition. The perturbed solutions were again projected back on to the reduced space spanned by the EOFs. Taken all together, the forcing response is expressed by an 15x3 matrix $\mathbf{F_R}$. The initial field in the reduced space, **X(0)**, was similarly constructed from the initial field of the full numerical simulation projected onto the reduced space spanned by the 18 EOFs.

The reduced model is then expressed as:

$$X(t+1) = M_R X(t) + F_R U(t) \tag{7}$$

where U is the forcing projected onto the reduced subspace used for the forcing.

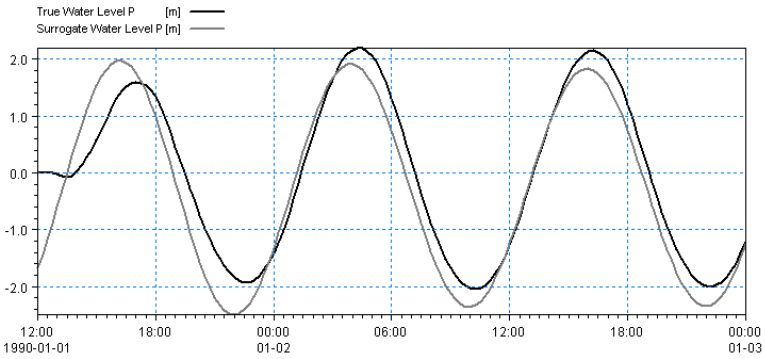


Figure 3 Time series comparison at station P. Black line is from the numerical hydrodynamic model MIKE 21, while the grey line is the result of the surrogate model

RESULTS

Realistic simulation

This part demonstrates the skill of the surrogate model for modeling the flow and water level for a reference simulation. Hence the water levels and fluxes predicted by the surrogate model are compared to those predicted by the complex numerical simulation. The comparison is shown in Figures 4, 5 and 6. Figure 4 shows an comparison of the water level time series in point P. Figures 5 and 6 show snapshots of water level for the true MIKE 21 and the surrogate model respectively. It is evedent that the main modes of variability are successfully captured by the surrogate model.

Figure 6 Snapshot of water level from the Surrogate model

Monte Carlo simulations

This part demonstrates the skill of the surrogate model for predicting the model uncertainty in response to uncertain open boundary conditions and wind forcing. Monte Carlo simulations was performed for both a stochastic version of the numerical hydrodynamic model MIKE 21 and the surrogate model. A steady reference solution was

used and AR(1) noise processes of the open boundary and wind was imposed with the same parameters. Figure 7 shows the results of the traditional and surrogate approach. The resemblance is remarkable considering that only 15 degrees of freedom was used in the surrogate modeling.

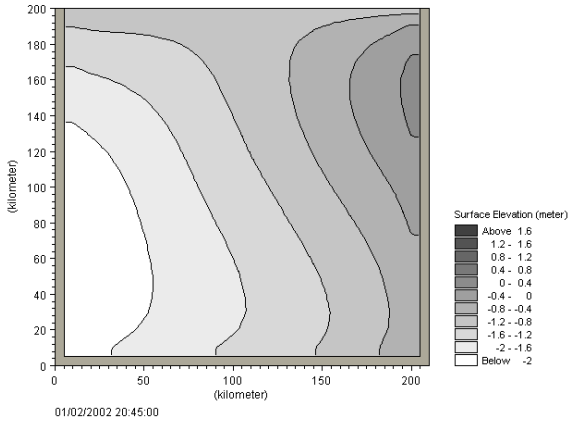


Figure 4 Snapshot of water level from the numerical hydrodynamic model MIKE 21

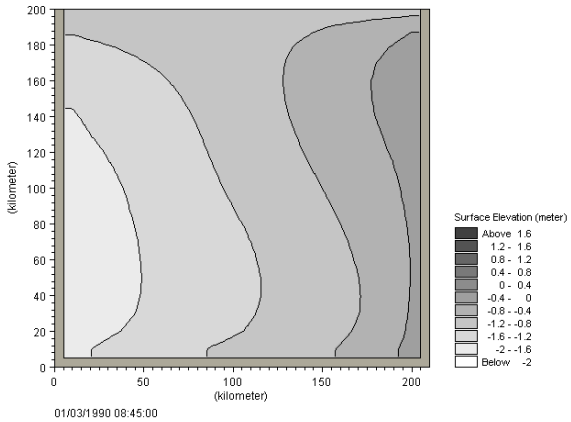


Figure 5 Snapshot of water level from the surrogate model

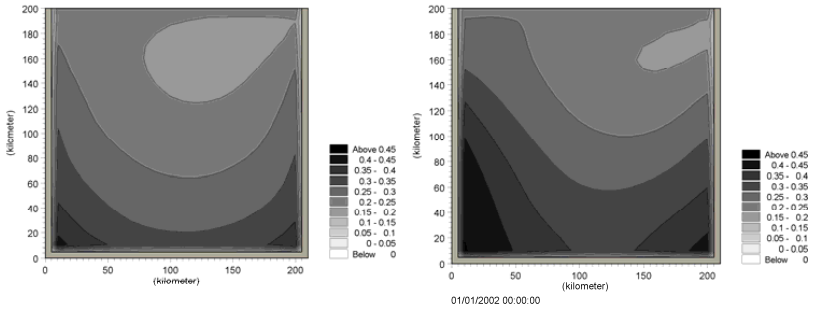


Figure 7 Comparison of water level standard deviation derived from Monte Carlo simulations of MIKE 21 (left) and the surrogate model (right)

CONCLUSIONS

A combined model reduction and Taylor series expansion approach has been presented for creating time efficient approximate models of complex and expensive dynamical models. This technique was demonstrated to successfully capture the dominating modes in a reference simulation and to provide a good estimate of the standard deviation of water levels in a Monte Carlo simulation.

It is interesting to note how few eigenmodes (5 for each variable) that are sufficient to give these results. However, in more realistic situations with complex bathymetry and local physical phenomena a much larger subspace should be retained. To achive good results in such cases, a continuing effort must be devoted to a number of issues such as 1) selection of snapshot input data to the EOF analysis, 2) investigation of various expansion techniques and their suitability for deriving efficient solution for particular modeled phenomena, 3) multiple surrogate models for different expansion points and 4) trade-off between size of reduced space, expansion technique and order versus computational time.

REFERENCES

[1] Vermeulen, P.T.M. and A.W. Heemink, "Efficient Implementation of the Adjoint method Using ModelReduction", *Monthly Waether Review*, (2005), 19p

[2] Holmes, P., J.L. Lumley, and G. Berkooz, "*Turbulence, Coherent Structures, Dynamical Systems and Symmetry"*, Cambridge Univ. Press., 1996

[3] Fukumori, I. R. Raghunath, L.-L. Fu and Y. Chao, "Assimilation of TOPEX/Poseidon altimeter data into a global ocean circulation model: How good are the results?", *J. of Geoph. Res. Oceans*, 104, C11 (1999), 19p

APPENDIX E

Paper E

Accepted for publication as IMM-Technical Report-2009-02

Particle filters
with Applications

Jan Frydendall

May 29, 2009

DTU Informatics

Contents

1 The state space model

The type of filtering that we are interested is state space filtering. Hence, we are given an arbitrary state space model with a set of observations. The task is now to construct a mechanism that would enable us to do estimations of the state space model parameters and reconstruction of the states. In our setting we are mostly interested in non linear state space models. The notion on linear state space models is well covered in [11]. The general form of the state space model is the following [14]

$$\mathbf{x}_{t+1} = \mathbf{f}_t(\mathbf{x}_t, \mathbf{v}_t) \tag{1}$$

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t, \mathbf{w}_t), \tag{2}$$

where $\mathbf{f}_t : \mathcal{R}^n \times \mathcal{R}^m \to \mathcal{R}^n$ is the model transition operator and $\mathbf{v}_t \in \mathcal{R}^m$ is a white noise process that is not dependent on the past and current states. The $\mathbf{h}_t : \mathcal{R}^p \times \mathcal{R}^r \to \mathcal{R}^r$ is the observation operator that relates the states to the observations. The observations noise $\mathbf{w}_t \in \mathcal{R}^r$ is also and white noise process that is not dependent on the past and current states and the system noise. It is assumed that the PDF of $\mathbf{v}_t$ and $\mathbf{w}_t$ is known and that we also know the initial distribution of $p(\mathbf{x}_1|\mathbf{y}_0) = p(\mathbf{x}_0)$ together with transition and observation operator for all $t \in \{0, \cdots, T\}$. The task is to construct the PDF of the current state $\mathbf{x}_t$ given all the information avaliable to us $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Much as we have done with Kalman filter and the Extended Kalman filter in [11, 12] we will do the construction in two steps, a prediction and a update step. Assume that we have the PDF $p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$ at time step $t-1$ we can now construct the $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t-1})$ with the transition operator,

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{1:t-1}, \tag{3}$$

where $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the transition probability that is generated from the transition model with known $\mathbf{v}_{t-1}$. The transition PDF is generated from the Markov state model (1) and the known PDF of $\mathbf{v}_{t-1}$, hence

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{v}_{t-1})p(\mathbf{v}_{t-1}|\mathbf{x}_{t-1})d\mathbf{v}_{t-1}, \tag{4}$$

by assumption we have that $p(\mathbf{v}_{t-1}) = p(\mathbf{v}_{t-1}|\mathbf{x}_{t-1})$, thus we get

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \int \delta(\mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{v}_{t-1})p(\mathbf{v}_{t-1})d\mathbf{v}_{t-1}, \tag{5}$$

the delta function $\delta(\cdot)$ indicates that we do not known $\mathbf{x}_{t-1}$ and $\mathbf{v}_{t-1}$ explicitly. If we did we could easily get $\mathbf{x}_t$ from (1) [14]. When the measurements becomes avaliable we can update the prior via Bayes rule [4],

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{0:t-1})}{\int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{0:t-1})d\mathbf{x}_t} \tag{6}$$

The likelihood PDF, $p(\mathbf{y}_t|\mathbf{x}_t)$ is defined by the observation operator and the known distribution $\mathbf{w}_t$,

$$p(\mathbf{y}_t|\mathbf{x}_t) = \int \delta(\mathbf{y}_t - \mathbf{h}_t(\mathbf{x}_t, \mathbf{w}_t))p(\mathbf{w}_t)d\mathbf{w}_t. \tag{7}$$

The equation (6) is used to update the prediction prior (3) when new measurements $\mathbf{y}_t$ become avaliable. This is required if we want to obtain the posterior of the states. The equation (3) and (6) is the solution of the Bayes recursive estimation problem [14]. The only known analytical solution to these equations is the Kalman filter, if we assume that the our state space model is linear and the noise process are normal distributed.

An example of a nonlinare state space model is

$$x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{1 + x_{t-1}^2} + 8cos(1.2(t-1)) + v_t \tag{8}$$

$$y_t = \frac{x_t^2}{20} + w_t, \tag{9}$$

where $v_t \sim \mathcal{N}(0, \sigma_v^2)$ and $w_t \sim \mathcal{N}(0, \sigma_w^2)$ are normal distributed PDF with known variance. This model is widely used in literature [16, 14, 4],when it comes to particle filtering. In our example we will use the following initial conditions, $\sigma_v^2 = 10$, $\sigma_w^2 = 1$ and $x_0 \sim \mathcal{N}(0, 10)$. In figure 1 is the state space model represented.

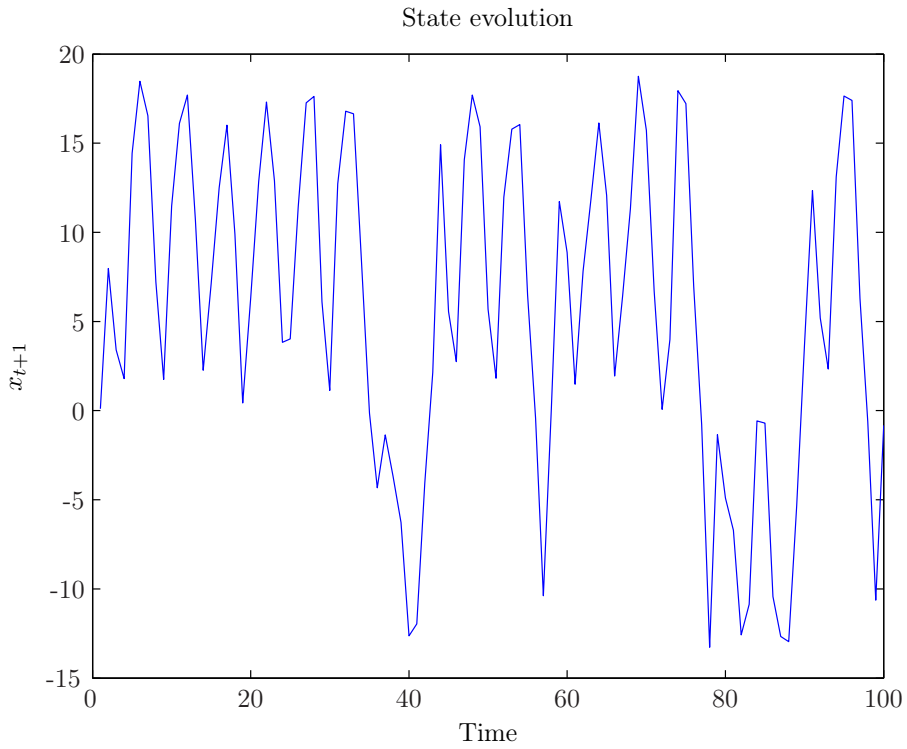


Figure 1: The simulate states of the model (8)

2 The preliminaries

In order to take full advantage of the particle filters, we have to address the two most important key elements in the particle filter cocktail. Namely the Monte Carlo interpretation and the Importance sampling steps. First we will address the Monte Carlo requirement.

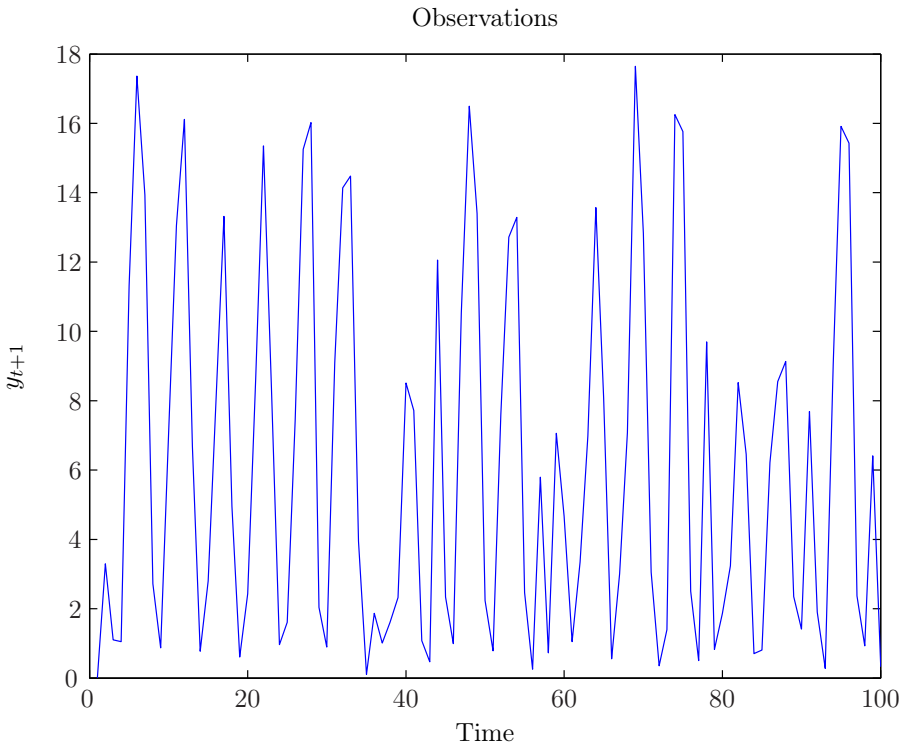


Figure 2: The observation from the state evolution through the observation operator (9)

2.1 Perfect Monte Carlo Simulation

To put Monte Carlo simulation in lay man terms; Monte Carlo simulation is just a another way of interprete integrals as sums. However, this loose definition will not do us any good mathematically. In order to get a rigorous mathematical definition we assume the following. Assume that we have some function $f(x) \in \mathcal{R}$ that we would like to know the expectation. We use the definition of the expectation by assuming that we have a given

probability distribution function (pdf) $p(x, y) \in \mathcal{R}$. The expectation can be written as

$$E[f(x)] = \int_{-\infty}^{\infty} f(x)p(x, y)dx, \tag{10}$$

most often the integral can not be solve analytically and therefore we most find another way. One way is the Monte Carlo simulation, Lets assume that we have N independent and identically distributed (i.i.d.) random samples from $\{\mathbf{x}_{0:t}^{(i)}; i = 1, \ldots, N\}$ according to our pdf $p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t})$. As in the book by [4] we will define the empirical distribution as a sum of dirac delta functions,

$$p_N(d\mathbf{x}_{0:t}, \mathbf{y}_{0:t}) = \sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^i}(d\mathbf{x}_{0:t}), \tag{11}$$

where $\delta_{\mathbf{x}_{0:t}^i}(d\mathbf{x}_{0:t})$ is the probability mass located in $\mathbf{x}_{0:t}^i$. The $d\mathbf{x}_{0:t}$ just the denotes that we are in continuous formulation and that we can not specify the exact location of the probability mass, so we only specify the probability mass in a vicinity of $\mathbf{x}_{0:t}^i$ by make a small sphere around it. With this definition we able to estimate the expectation (10) as

$$\overline{E[\mathbf{f}_t(\mathbf{x}_{0:t})]} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{f}(\mathbf{x}_{0:t}^i), \tag{12}$$

where $\mathbf{f}_t$ is the function to be estimated and for generality we have made it time depending and multidimensional. According to law of large numbers, the expectation (12) will converge almost surely to (10), i.e $\overline{E[\mathbf{f}_t(\mathbf{x}_{0:t})]} \xrightarrow[N\to\infty]{a.s.} E[\mathbf{f}_t(\mathbf{x}_{0:t})]$ and if the posterior variance of $\mathbf{f}_t(\mathbf{x}_{0:t})$ is bounded i.e. $\sigma_{\mathbf{f}_t}^2 < \infty$ we then have from the central limit theorem that [15, 4]

$$\sqrt{N}\left(\overline{E[\mathbf{f}_t(\mathbf{x}_{0:t})]} - E[\mathbf{f}_t(\mathbf{x}_{0:t})]\right) \underset{N\to\infty}{\Longrightarrow} \mathcal{N}(0, \sigma_{f_t}^2),$$

where $\underset{N\to\infty}{\Longrightarrow}$ denotes converges in distribution.

2.2 Bayesian Importance Sampling

Here we will develop the idea of Importance sampling, which in short is to find a proper proposal distribution by considering a clever scaling distribution. Consider again the expectation of f_t

$$\begin{aligned} E[\mathbf{f}_t(\mathbf{x}_{0:t})] &= \int \mathbf{f}_t(\mathbf{x}_{0:t}) \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\ &= \int \mathbf{f}_t(\mathbf{x}_{0:t}) \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\ &= \int \mathbf{f}_t(\mathbf{x}_{0:t}) \frac{\omega_t(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})} q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}, \end{aligned}$$

where we $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ is the new proposal distribution and $\omega_t(\mathbf{x}_{0:t})$ is the importance weights

$$\omega_t(\mathbf{x}_{0:t}) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}. \tag{13}$$

The distribution $p(\mathbf{y}_{1:t})$ is not depending on $\mathbf{x}_{0:t}$ and therefore it can be brought outside the integral. We can again use that the distribution is the marginalization of $p(\mathbf{y}_{1:t}) = \int p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})d\mathbf{x}_{0:t}$ and rewrite the expectation

$$\begin{aligned} E[\mathbf{f}_t(\mathbf{x}_{0:t})] &= \frac{1}{p(\mathbf{y}_{1:t})} \int \mathbf{f}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\ &= \frac{\int \mathbf{f}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})\frac{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}d\mathbf{x}_{0:t}} \\ &= \frac{\int \mathbf{f}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}}{\int \omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}} \\ &= \frac{E_{q(\cdot|\mathbf{y}_{1:t})}(\omega_t(\mathbf{x}_{0:t})\mathbf{f}_t(\mathbf{x}_{0:t}))}{E_{q(\cdot|\mathbf{y}_{1:t})}(\omega_t(\mathbf{x}_{0:t}))}, \end{aligned}$$

where $E_{q(\cdot|\mathbf{y}_{1:t})}$ denotes the expectation of $\omega_t(\mathbf{x}_{0:t})$ with respect to the proposal distribution $q(\cdot|\mathbf{y}_{1:t})$. With this in mind we can now express the

expectation as

$$E[\mathbf{f}_t(\mathbf{x}_{0:t})] = \frac{\frac{1}{N}\sum_{i=1}^{N}\mathbf{f}_t(\mathbf{x}_{0:t}^i)\omega_t(\mathbf{x}_{0:t}^i)}{\frac{1}{N}\sum_{i=1}^{N}\omega_t(\mathbf{x}_{0:t}^i)} \tag{14}$$

$$= \sum_{i=1}^{N}\mathbf{f}_t(\mathbf{x}_{0:t}^i)\tilde{\omega}_t(\mathbf{x}_{0:t}^i), \tag{15}$$

where $\tilde{\omega}_t^i$ are the normalized importance weights,

$$\tilde{\omega}_t^i = \frac{\omega_t^i}{\sum_{j=1}^{N}\omega_t^j}. \tag{16}$$

The expectation (14) is biased as long as N is finite. However, from the law of strong numbers the estimate is asymptotically unbiased, for a good discussion of this look in [6]. As N tens to infinity, the posterior density function can be approximated arbitrarily well by the point-mass estimate.

$$\hat{p}(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N}\tilde{\omega}_t(\mathbf{x}_{0:t}^i)\delta_{\mathbf{x}_{0:t}^i}(d\mathbf{x}_{0:t}). \tag{17}$$

3 Particle Filters

After all the sung and dance in the previous section we now able to formulate the particle filters. The first an most straight forward particle filter is the one called the Sequential Importance Sampling (SIS). The idea is to draw samples from our prior distribution and then assign an importance weights to each particle at every time step. Put into a more formal frame, we first have to make some assumptions on the Importance weights. In order to sample from the proposal distribution $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, we have to assume that we can compute the sequential estimate of the posterior distribution at time t without modifying the previously simulate states $\mathbf{x}_{0:t-1}$ [15]. Thus, the following distribution can be used

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}). \tag{18}$$

Hence the current state is independent of the future observations. Along with the Markov properties of the states

$$p(\mathbf{x}_{0:t}) = p(\mathbf{x}_0) \prod_{j=1}^{t} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \tag{19}$$

and that the observations are conditionally independent given the states

$$p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t}) = \prod_{j=1}^{t} p(\mathbf{y}_j|\mathbf{x}_j) \tag{20}$$

If we now apply the above assumption onto the Importance weights (13) we can establish the following recursive importance weights

$$\begin{aligned} \omega_t &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})} \\ &= \omega_{t-1} \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{x}_{0:t-1})p(\mathbf{x}_{0:t-1})} \frac{1}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})} \\ &= \omega_{t-1} \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})} \end{aligned} \tag{21}$$

With the above recursive importance weights we can now at each time step generate a new importance weight, i.e. a new state reconstruction, since we are able to calculate the likehood $p(\mathbf{y}_t|\mathbf{x}_t)$ and the transition probabilities $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, given that we have a prober proposal distribution $q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})$.

Until now we have just stated that the proposal weights are something that is essential in the this estimation approach. However, finding the prober proposal distribution is a very difficult task. Much of the current research is devoted to finding prober proposal distribution or avoiding them.

3.1 Proposal distributions

The most critical ingredient in the particle filters that uses importance sampling are the proposal distributions. The foremost property of the proposal

distribution is to minimize the variance of the importance weights conditional on $\mathbf{x}_{0:t-1}$ and $\mathbf{y}_{1:t}$ [5]. The prober choice is then the distribution $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})$, that minimizes the variance of the importance weights. However, this result does not do much for us, it is only of theoretical interest. Among practitioners the following proposal distribution is popular [14]

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{22}$$

where the proposal distribution now is distributed as the transition prior. Although we are given up the requirement of the minimizing variance of the importance weights, we have on the other hand got a proposal distribution which most easier to implement and to sample from. If we substitute the (22) into the (21) we get a very nice result

$$\omega_t = w_{t-1}\frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_t|\mathbf{x}_{t-1})} = w_{t-1}p(\mathbf{y}_t|\mathbf{x}_t). \tag{23}$$

In the next section we will give some examples of the SIS and the other variant called the SIR.

4 SIS

If we use the idea of importance sampling (23) from the previous section we can formulate the two basic particle filters. The first filter is the SIS filter and the generic algorithm is shown below in algorithm (1).

When we use the transition prior as proposal distribution the filters are often called *Bootstrap* filters.

The SIS filters has one major problem that over shadows everything else. The key idea with the SIS filters is to select the particles that have the most probable outcome compared to the observation given the predictions. With this algorithm we are thinning out the particles that do not represent the state and observations. When this thinning is applied at every time step we more or less thin out all particles except one. This problem is

1: **procedure** SIS
2: **Initialization,** $t = 0$
For $i = 1, \cdots, N$, sample $\mathbf{x}_0^i \sim p(\mathbf{x}_0)$ and set $t = 1$.
3: **Importance sampling step**
For $i = 1, \cdots, N$, sample $\mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and set $\tilde{\mathbf{x}}_{0:t}^i = \left(\mathbf{x}_{0:t-1}^i, \tilde{\mathbf{x}}_t\right)$.
For $i = 1, \cdots, N$, evaluate the importance weights $\omega_t = p(\mathbf{y}_t|\mathbf{x}_t)$
Normalize the importance weights.
Set $t \to t + 1$ and go to step 3
4: **end procedure**

Algoritme 1: The generic SIS algorithm

known as degeneracy and can be illustrated in the figure (3). Put in more mathematically formalism we say that the variance of the weights increases over time. If we again look at the definition of the importance (13) weights we have,

$$\begin{aligned} \omega_t(\mathbf{x}_{0:t}) &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &= \frac{p(\mathbf{y}_{1:t}, \mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &= \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})p(\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &\propto \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}, \end{aligned} \tag{24}$$

where we have used that $p(\mathbf{y}_{1:t})$ is a constant. The ratio (24) is called the importance ratio and it can be shown that the variance of this ration will increase over time. This has been done by ([10, 5]). Following the argument from [15] we want to have that the proposal and the posterior density to be close, i.e. that is the proposal distribution has full support over the true posterior density. Taking the expectation of the importance ratio with

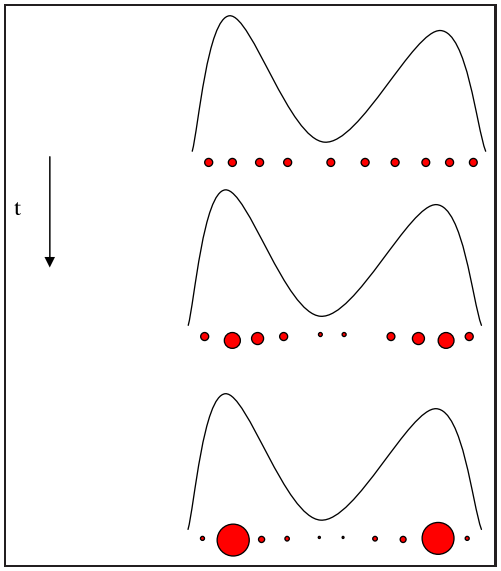


Figure 3: The degeneracy problem of the SIS filters. At each time step the most probable particles are given an weight that represents the probability that prediction is close to the observations.

respect to the proposal distribution

$$E_{q(\cdot|\mathbf{y}_{1:t})}\left(\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right) = 1 \tag{25}$$

and the variance

$$var_{q(\cdot|\mathbf{y}_{1:t})}\left(\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right) = E_{q(\cdot|\mathbf{y}_{1:t})}\left(\left(\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} - 1\right)^2\right), \tag{26}$$

this gives an indication of that we want the variance of the importance weights to be close to zero in order to get good estimates. However, when the variance of the weights increase over time we get inaccurate estimations.

5 SIR

When doing filtration with SIS algorithm one has to start out with a very large numbers of particle in order make sure that some of the particle survives. In order to reduce the particle set size and thereby reducing the computational time, we will consider the SIR filter. The idea with the SIR filter is to resample the particles at each time step. We still uses the importance sampling algorithm, however, after we have assign weights to the particles will only keep the particles that have more weight then $1/N$ and then resample the particles that has less weights from the surviving particles.

Assume that to each particle $\mathbf{x}_{0:t}^i$ we can assign an weight $N_i \in \mathcal{N}$ such that $\sum_{i=1}^N N_i = N$ and that we can rewrite (17) as

$$P_N(d\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} N_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}) \tag{27}$$

The above assumption leads to the following interpretation of the weights. At each time step we assign the particles importance weights and then we resample the particles such that after the resampling step all the particles will have equal probability $1/N$. We do not change the number of particles in the set we only discard the ones that are unlikely and multiply the ones that survives such that the total number of particles are N. The SIR algorithm can also be written in a generic algorithm

5.1 Resampling

The selection of the particles and the resampling step can be done in many ways. Here we only discuss the very basic resampling algorithm, namely the multinomial sampling. We want to find a mapping from $\{\mathbf{x}_{0:t}^i, \tilde{\omega}_t^i\} \rightarrow \{\mathbf{x}_{0:t}^i, N^{-1}\}$. The mapping is given in [14] and can be written as

1: **procedure** SIR
2: **Initialization,** $t = 0$
For $i = 1, \cdots, N$, sample $\mathbf{x}_0^i \sim p(\mathbf{x}_0)$ and set $t = 1$.
3: **Importance sampling step**
For $i = 1, \cdots, N$, sample $\mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and set $\tilde{\mathbf{x}}_{0:t}^i = \left(\mathbf{x}_{0:t-1}^i, \tilde{\mathbf{x}}_t\right)$.
For $i = 1, \cdots, N$, evaluate the importance weights $\omega_t = p(\mathbf{y}_t|\mathbf{x}_t)$
Normalize the importance weights.
4: **Resampling step**
Multiply/Suppress samples $\tilde{\mathbf{x}}_{0:t}^i$ with high/low importance weights $\tilde{\omega}_t^i$, respectively,
to obtain N random samples $\mathbf{x}_{0:t}^i$ approximately distributed to $p(\mathbf{y}_t|\mathbf{x}_t)$
For $i = 1, \cdots, N$ set $\omega_t^i = \tilde{\omega}_t^i = \frac{1}{N}$
Set $t \to t + 1$ and go to step 3
5: **end procedure**

Algoritme 2: The generic SIR algorithm

$$\sum_{j=1}^{M-1} \tilde{\omega}_t^j < u_i \leq \sum_{j=1}^{M} \tilde{\omega}_t^j, \tag{28}$$

where u_i is a random sampling from the uniform distribution $\mathcal{U}(0,1]$. This procedure is repeated for $i = 1, \cdots, N$. This algorithm can be written in generic form

```
1: procedure MULTINOMIAL
2:     For i = 1, ..., N, sample u^i ~ U(0,1]
3:     calculate the discrete c.d.f. of u_i and ω̃_t^i
4:     set j = 1 and i = 1
5:     while j ≤ N do
6:         if c.d.f.(u_i) - c.d.f.(ω̃_t^i) ≤ 0 then
7:             idx(j) = i
8:             j = j + 1
9:         else
10:            i = i + 1
11:        end if
12:    end while
13: end procedure
```

Algoritme 3: The generic multinomial sampling algorithm

where $idx(i)$ is the index function. The graphical interpretation of the algorithm is given in figure (4).

The c.d.f. of u_i is seen as the strait line in the figure (4) and the step line is the c.d.f. of $\tilde{\omega}_t^i$. At the top of the figure is the particle weights shown. When the particles have great mass then the step function is above the strait line and the particles are preserved and multiply to the locations where the particles have little mass. When the particles have small mass the step function is below the strait line and the particular particle is killed and gets resampled from one of the particles that have great mass. There exist of corse more resampling algorithms in literature see for example [3, 7] for a survey of the most common used.

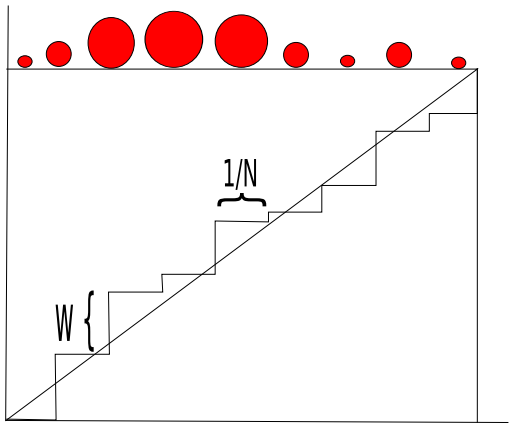


Figure 4: The graphical interpretation of the algorithm (3).

6 Application

In this section we will give examples of the particle filters. We will use the state space model (1) and (2) as the test bed of the filtration. First we will try the SIS filter. In the filtration we will use 500 particles and the noise is assumed to be normal distributed. Thus, the model noise $Q \sim \mathcal{N}(0, 10)$ and the observation noise $R \sim \mathcal{N}(0, 1)$. The observation operator H is given as $H = y^2$, where y is the input from the predictions. Hence, the filter does not know explicitly if the sign of the states and therefore around zero the filter will have trouble with the estimation since the there is no indication of the dynamics of the state. The likelihood function $p(\mathbf{y_t}|\mathbf{x_t})$ will be given as an Gaussian bell

$$\omega_t^i = \frac{\exp(-0.5((\mathbf{y}_t^i - \mathbf{H}\mathbf{x}_t^i)^T \mathbf{R}^{-1}(\mathbf{y}_t^i - \mathbf{H}\mathbf{x}_t^i))}{\sum_{j=1}^N \exp(-0.5((\mathbf{y}_t^j - \mathbf{H}\mathbf{x}_t^j)^T \mathbf{R}^{-1}(\mathbf{y}_t^j - \mathbf{H}\mathbf{x}_t^j)))}. \tag{29}$$

In the figure (5) the reconstruction of the states are shown and below is the scatter plot of the reconstructed states and the observations. It is not obvious that this filtration is very good since from the plot we can see that

the reconstruction is following the observations and the scatter plot is also close to the strait line. However, when we consider the the effective particle size i.e. the inverse of the variance of the importance weights,figure (6).

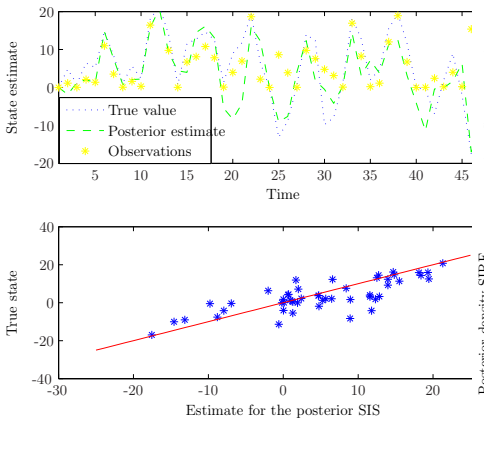


Figure 5: The reconstruction of the states with the SIS algorithm

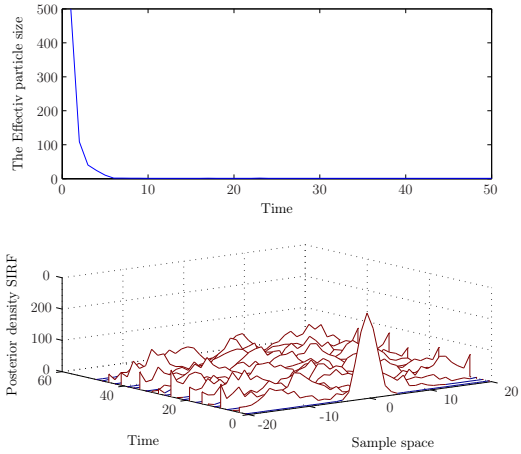


Figure 6: The effective particle size and the posterior densities from the SIS estimation

We can see that the particle set is reduced to effectively one single particle. This means that the filtration is only consisting of one realization of the process and therefore we can not assume that we will converge to the true states. This can also be seen from the density plot in the same figure. The densities starts out as a concentration around the prior, however, as the simulation continues the particle sizes is reduced and the densities get smoother out and will not carry any useful information trough to the prediction.

As mention earlier, one way of taking the degeneracy into account is to include a resampling step after each importance sampling step to keep the particles alive.

In the figure (7) we can see that the reconstruction is better then the one

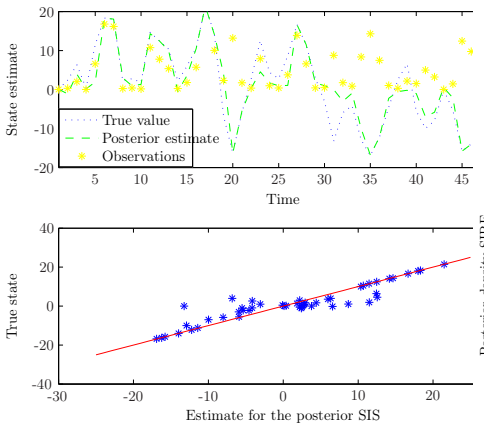


Figure 7: The reconstruction of the states with the resampling at every time step

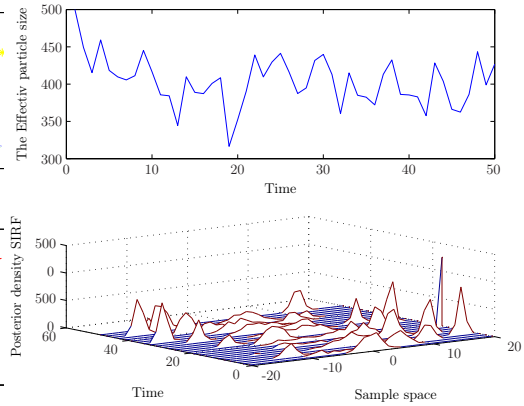


Figure 8: The effective particle size and the posterior densities with the resampling of the particles

from (5). (We are using the same seed in the random number generator). Remember that the observations in the figures are generated trough the observation operator and therefore we can not compared them directly to the states.

Looking at the effective particle size figure (8) we can that the resampling step is keeping the particle set well stirred and the all the particles carry information. This is also evident from the posterior density plot. We see that since we have a much effective particle set the densities will carry information thorough to the next prediction step. In this plot it is also clear that the posterior densities are not Gaussians and the strength of the particle filters should be obvious compared to the Extended Kalman Filter.

However, the resampling step is increasing the variance of the posterior estimates and therefore it should not be performed unless it is necessary. Therefore it is suggested that we use a form of threshold sampling for example $N_eff \leq \frac{2}{3}N$, where $N_eff = (\sum_{j=1}^{N}(\omega_t^j)^2)^{-1}$. This means that we use

the SIS filter at every time and when the effective particle size is reduced to less then the threshold then we perform the resampling step. Put into the generic algorithm form The estimation with SIS with threshold sampling

1: **procedure** SIS WITH THRESHOLD SAMPLING
2: **Initialization,** $t = 0$
For $i = 1, \cdots, N$, sample $\mathbf{x}_0^i \sim p(\mathbf{x}_0)$ and set $t = 1$.
3: **Importance sampling step**
For $i = 1, \cdots, N$, sample $\mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and set $\tilde{\mathbf{x}}_{0:t}^i = \left(\mathbf{x}_{0:t-1}^i, \tilde{\mathbf{x}}_t\right)$.
For $i = 1, \cdots, N$, evaluate the importance weights $\omega_t = p(\mathbf{y}_t|\mathbf{x}_t)$
Normalize the importance weights.
4: **if** $N_eff \leq \beta N$ **then**
5: **Perform the resampling step**
6: Multiply/Suppress samples $\tilde{\mathbf{x}}_{0:t}^i$ with high/low importance weights $\tilde{\omega}_t^i$, respectively,
to obtain N random samples $\mathbf{x}_{0:t}^i$ approximately distributed to $p(\mathbf{y}_t|\mathbf{x}_t)$
For $i = 1, \cdots, N$ set $\omega_t^i = \tilde{\omega}_t^i = \frac{1}{N}$
7: **end if**
Set $t \to t + 1$ and go to step 3
8: **end procedure**

Algoritme 4: The generic SIS with threshold sampling algorithm

gives the same result as with SIR. However, we have made sure that with this approach has some sort of variance reduction on the posterior estimates. The estimation with SIS with threshold sampling can be seen in the figures (9) and (10).

The final comparison is done with run two independent filtration with $N = 250$ and $N = 500$ with same noise assumptions. In table (6) the mean and the variance and the root mean squared error of the 1-step prediction error is shown. (Runs with higher particle numbers has also been conducted, however, results showed that the estimation had converge i.e. there was no

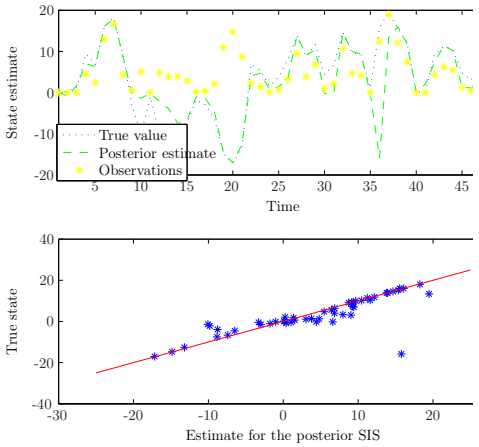


Figure 9: The reconstruction of the states with the threshold sampling

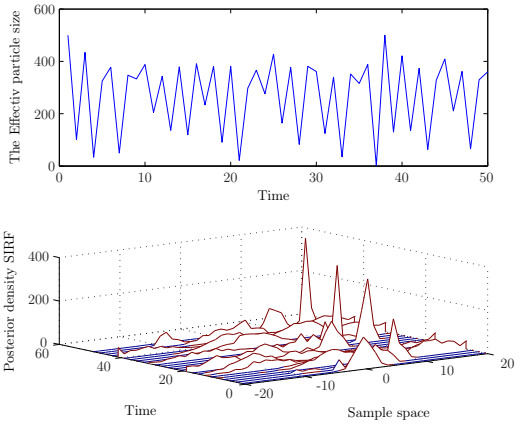


Figure 10: The effective particle size and the posterior densities from the threshold sampling

improvement in the bias, variance and rms).

Final words on importance particle filtering, the trouble with particle filters is that on never know how many particles to use. In our simple state space model we only had to reconstructed one state, however, if the state space model goes to higher dimensions the number of particles could very well increase by a factor 1000. There are many reference in litterateur on how to reduce the particle size and how to implement them. The auxiliary SIR and the kalman filter hybrids as being the most common. However, when we straying away from the Kalman Filter and are trying with other non-parametric filters, we should be very aware of the *no free lunch* theorem. The theorem states that may the SIR filter is better on this type of model that we have uses here, however, could very well be that is the other way around with another model [2]. With this in mind the particle filters is very easy and a very good estimator for non-linear models. With growing computational power the particle filters are getting better and better.

$N = 1000$	SIS		SIR		$Neff \leq 2/3N$	
N	250	500	250	500	250	500
mean	-2.133	-1.676	-0.514	-1.012	-1.008	-0.821
var	120.086	34.556	32.068	10.935	18.430	9.454
rms	11.164	6.113	6.051	3.708	4.7939	3.512

Table 1: The statistics of two simulation with particle filters, for respectively $N = 250$ and $N = 500$

7 Particle smoothers

In this section the notion of smoothing will be investigated. In this section two particle smoothers will be discussed.

1. The Forward-Backward Smoother (FBS)
2. The two filter smoother (TFS)

7.1 The Forward-Backward Smoother

The FBS is the simplest to construct it relies on a forward filtering in time up till the desired time to obtain the marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. Then a backward sweep is thourgh the data set is conducted to modify the importance weights so that they now represent the smoothed distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. The algorithm for the FBS is simple and intuitive However, the FBS relies on that the filter distribution has support where the smoothed density is significant [9].

The quest is to construct the marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ from the for-

ward filter and the backward recursion.

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:T}) &= \int p(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{y}_{1:T}) d\mathbf{x}_{t+1} \\ &= \int p(\mathbf{x}_{t+1}|\mathbf{y}_{1:T}) p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}) d\mathbf{x}_{t+1} \\ &= p(\mathbf{x}_t|\mathbf{y}_{1:t}) \int \frac{p(\mathbf{x}_{t+1}|\mathbf{y}_{1:T}) p(\mathbf{x}_{t+1}|\mathbf{x}_t)}{\int p(\mathbf{x}_{t+1}|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t}) d\mathbf{x}_t} d\mathbf{x}_{t+1}, \end{aligned} \tag{30}$$

where

- $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is the filtered density
- $p(\mathbf{x}_{t+1}|\mathbf{y}_{1:T})$ is the smoothed density
- $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ is the dynamics of the model
- $\int p(\mathbf{x}_{t+1}|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t}) d\mathbf{x}_t$ is the state prediction

The recursion is approximated in the usual way by defining the an empirical distribution [9]

$$\hat{p}(d\mathbf{x}_t|\mathbf{y}_{1:T}) = \sum_{i=1}^{N} \omega_{t|T}^i \delta_{\mathbf{x}_t^i} d\mathbf{x}_t \tag{31}$$

and inserted into (30) thus we get the empirical approximation of (30)

$$\omega_{t|T}^i = \omega_{t+1|T}^i \left[\sum_{j=1}^{N} \omega_{t+1|T}^j \frac{p(\mathbf{x}_{t+1}^j|\mathbf{x}_t^i)}{\sum_{k=1}^{N} \omega_t^k p(\mathbf{x}_{t+1}^j|\mathbf{x}_t^k)} \right]. \tag{32}$$

The (32) is of the order $\mathcal{O}(N^2)$ by noticing that the denominator can be calculated inpendently from i for each j. The algorithm for the FBS can be seen in algorithm 5 In figure (7.1) the marginal distribution of the FBS is shown. The difference between the forward filter and the FBS is not significant. The figure also underlines the problem with FBS. The smoother can not change the support of the particles if there located in the wrong part of the state space. No new information is added to the smoohter, so the only option is to relocate the partices in the given support.

1: **procedure** FBS
2: *Filtering* For $t = 1, \cdots, T$, perform the particle filtering to obtain the weighted measure $\{\mathbf{x}_t^i, \omega_t^i\}_{i=1}^N$
3: *Initialization* For $i = 1, \cdots, N$, set $\omega_{T|T}^i = \omega_T^i$
4: *Backward recursion* For $t = T-1, \cdots, 1$ and $i = 1, \cdots, N$ evaluate $\omega_{t|T}^i = \omega_{t+1|T}^i \left[\sum_{j=1}^N \omega_{t+1|T}^j \frac{p(\mathbf{x}_{t+1}^j|\mathbf{x}_t^i)}{\sum_{k=1}^N \omega_t^k p(\mathbf{x}_{t+1}^j|\mathbf{x}_t^k)} \right]$
5: **end procedure**

Algoritme 5: The generic Forward-Backward smoother

7.2 The two filter smoother

The marginal smoothed posterior distribution can be computed by combining the output of tow independent filters [1]. The two filters that is need is first the normal particle filter and a filter that runs backward in time. The normal particle filter is just any one of the SMC filters that calculates $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$. The backward filter our barckward information filters is due to [13] and dates back to 1966. The backward filter calculates $p(\mathbf{y}_{t:T}|\mathbf{x}_t)$ backward in time. Combining the forward and backward filter we can obtain the smoothed marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:T})$, hence

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:T}) &= p(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \mathbf{y}_{t:T}) \\ &= \frac{p(\mathbf{x}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{t:T}|\mathbf{y}_{1:t-1}), \mathbf{x}_t)}{p(\mathbf{y}_{t:T}|\mathbf{y}_{1:t-1})} \\ &\propto p(\mathbf{x}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{t:T}|\mathbf{x}_t) \\ &\propto p(\mathbf{x}_t|\mathbf{y}_{1:t})p(\mathbf{y}_{t:T}|\mathbf{x}_t) \end{aligned} \tag{33}$$

The last line of (33) is the TFS the first density is the bayesian filter and the second density is the backward filter. This definition the of the TFS the normal forward filtering backward smoothing is now reduced to a pure filtration assumption. The backward filter can be calculated trough the backward information filter [13]

$$p(\mathbf{y}_{t:T}|\mathbf{x}_t) = \int p(\mathbf{y}_{t+1:T}|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{y}_t|\mathbf{x}_t)d\mathbf{x}_{t+1} \tag{34}$$

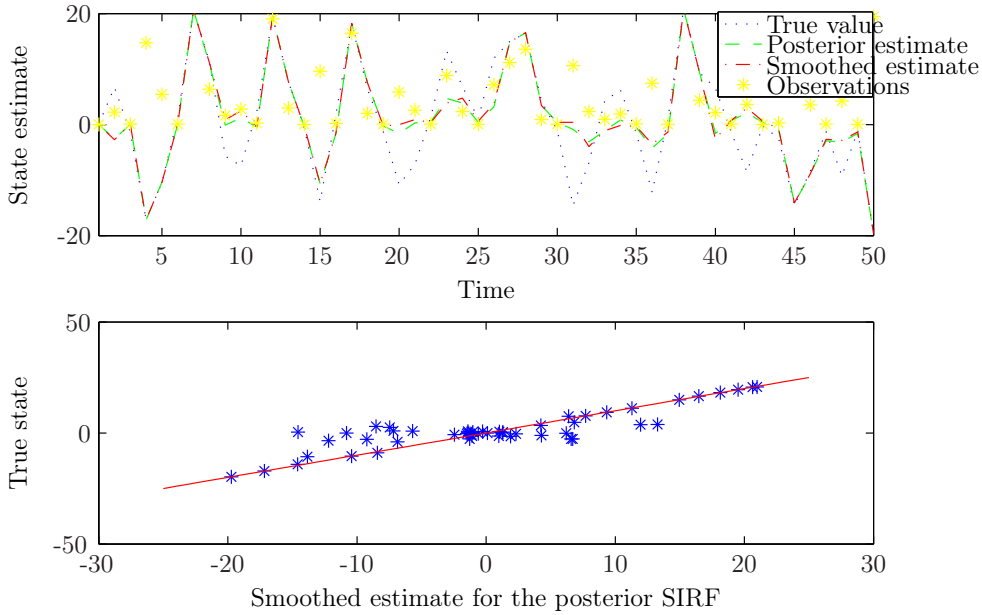


Figure 11: The smoothed estimate of the FBS and the scatter plot of the truth vs. the smoothed estimate

The problem with $p(\mathbf{y}_{t:T}|\mathbf{x}_t)$ is that is not a proper probability density in argument $\mathbf{x}_t$ and therefore the integral over $\mathbf{x}_t$ might not be finite [9]. Therefore normal sequential Monte Carlo approximation can not be used without making unrealistic assumptions, such as assuming $p(\mathbf{y}_{t:T}|\mathbf{x}_t) < \infty$ [8]. In order to make the TFS work for an arbitrary model the assumption derived in [1] and [9] will be used.

As state above the $p(\mathbf{y}_{t:T}|\mathbf{x}_t)$ is not a probability measure and therefore we can not apply the usually Monte Carlo methods since the constraint of these methods are that the probability densities have be finite. However, by introducing a clever artificial distribution over $\mathbf{x}_t$ with density $\gamma(\mathbf{x}_t)$ will ensure that the integral will be finite.

7.2.1 Artificial distribution

Let $\{\gamma_t(\mathbf{x}_t)\}$ be a sequence of probability distributions for $t = 1, \cdots, T$ such that for

$$p(\mathbf{y}_{t:T}|\mathbf{x}_t) > 0 \Rightarrow \gamma_t(\mathbf{x}_t) > 0 \tag{35}$$

and for the case $t = T$

$$\tilde{p}(\mathbf{x}_T|\mathbf{y}_t) = \frac{p(\mathbf{y}_T|\mathbf{x}_t)\gamma_T(\mathbf{x}_T)}{\int p(\mathbf{y}_T|\mathbf{x}_t)\gamma_T(\mathbf{x}_T)d\mathbf{x}_T}. \tag{36}$$

and for the case $t = \{2, \cdots, T-1\}$

$$\tilde{p}(\mathbf{x}_T|\mathbf{y}_t) = \frac{\gamma_T(\mathbf{x}_t)\prod_{i=t+1}^{T} p(\mathbf{x}_{i+1}|\mathbf{x}_i)\prod_{i=t}^{T} p(\mathbf{y}_i|\mathbf{x}_i)}{\int\cdots\int \gamma_T(\mathbf{x}_t)\prod_{i=t+1}^{T} p(\mathbf{x}_{i+1}|\mathbf{x}_i)\prod_{i=t}^{T} p(\mathbf{y}_i|\mathbf{x}_i)d\mathbf{x}_{t:T}} \tag{37}$$

Thus the general case for $t = \{1, \cdots, T\}$

$$p(\mathbf{y}_{t:T}|\mathbf{x}_t) = \tilde{p}(\mathbf{y}_{t:T})\frac{\tilde{p}(\mathbf{x}_T|\mathbf{y}_t)}{\gamma_T(\mathbf{x}_t)}, \tag{38}$$

where

$$\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t:T}) = \int\cdots\int \tilde{p}(\mathbf{x}_{t:T}|\mathbf{y}_{t:T})d\mathbf{x}_{t:T}. \tag{39}$$

The proof for $t = T$ follows strait from the definition. For $t = \{1, \cdots, T-1\}$

$$\begin{aligned}
p(\mathbf{y}_{t:T}|\mathbf{x}_t) &= \int \cdots \int p(\mathbf{y}_{t:T}, \mathbf{x}_{t+1:T}|\mathbf{x}_t) d\mathbf{x}_{t+1:T} \\
&= \int \cdots \int p(\mathbf{x}_{t+1:T}|\mathbf{x}_t) p(\mathbf{y}_{t:T}, \mathbf{x}_{t:T}) d\mathbf{x}_{t+1:T} \\
&= \int \cdots \int \prod_{i=t+1}^{T} p(\mathbf{x}_{i+1}|\mathbf{x}_i) \prod_{i=t}^{T} p(\mathbf{y}_i|\mathbf{x}_i) d\mathbf{x}_{t+1:T} \\
&= \int \cdots \int \frac{\gamma_T(\mathbf{x}_t)}{\gamma_T(\mathbf{x}_t)} \prod_{i=t+1}^{T} p(\mathbf{x}_{i+1}|\mathbf{x}_i) \prod_{i=t}^{T} p(\mathbf{y}_i|\mathbf{x}_i) d\mathbf{x}_{t+1:T} \\
&= \tilde{p}(\mathbf{y}_{t:T}) \int \cdots \int \frac{\tilde{p}(\mathbf{x}_{t:T}|\mathbf{y}_{t:T})}{\gamma_T(\mathbf{x}_t)} d\mathbf{x}_{t+1:T} \\
&= \tilde{p}(\mathbf{y}_{t:T}) \frac{\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t:T})}{\gamma_t(\mathbf{x}_t)}
\end{aligned}$$

7.2.2 The prediction and update steps

The prediction step with the backward filter is defined as.

$$\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t+1:T}) \triangleq \int \tilde{p}(\mathbf{x}_{t+1}|\mathbf{y}_{t+1:T}) \frac{p(\mathbf{x}_{t+1}|\mathbf{x}_t)\gamma_t(\mathbf{x}_t)}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1} \tag{40}$$

Again it has to be stessed that $\tilde{p}(\mathbf{x}_{t+1}|\mathbf{y}_{t+1:T})$ is not a probability measure if $\gamma_{t+1}(\mathbf{x}_{t+1}) \neq \int (\mathbf{x}_{t+1}|\mathbf{x}_t)\gamma_t(\mathbf{x}_t) d\mathbf{x}_t$. In order to have a finite intgral the artificial distribution $\{\gamma_t(\mathbf{x}_t)\}$ should selcted such that

$$\frac{p(\mathbf{x}_{t+1}|\mathbf{x}_t)}{\gamma_{t+1}(\mathbf{x}_{t+1})} < \infty, \tag{41}$$

for any $(\mathbf{x}_{t+1}, \mathbf{x}_t) \in \Omega$. Put in a more loosly tune we say that $\gamma_{t+1}(\mathbf{x}_{t+1})$ has have thicker tails than $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ for any $\mathbf{x}_t$ [1]. Before the update step

is defined the following needs to be calculated

$$\begin{aligned} p(\mathbf{y}_{t:T}|\mathbf{x}_t) &= \int p(\mathbf{y}_{t+1:T}|\mathbf{x}_t)p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{y}_t|\mathbf{x}_t)d\mathbf{x}_t \\ &= \int \frac{\tilde{p}(\mathbf{x}_{t+1}|\mathbf{y}_{t+1:T})}{\gamma_{t+1}(\mathbf{x}_{t+1})}p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{y}_t|\mathbf{x}_t)d\mathbf{x}_t \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_t)}{\gamma_t(\mathbf{x}_t)} \int \tilde{p}(\mathbf{x}_{t+1}|\mathbf{y}_{t+1:T})\frac{p(\mathbf{x}_{t+1}|\mathbf{x}_t)\gamma_t(\mathbf{x}_t)}{\gamma_{t+1}(\mathbf{x}_{t+1})}d\mathbf{x}_{t+1} \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_t)\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t+1:T})}{\gamma_t(\mathbf{x}_t)} \end{aligned} \tag{42}$$

with the above the update step can now be defined

$$\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t:T}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t+1:T})}{\int p(\mathbf{y}_t|\mathbf{x}_t)\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t+1:T})d\mathbf{x}_t}, \tag{43}$$

which has been renormalized to be a probability measure. The $\gamma_t(\mathbf{x}_t)$ in the denomenator of the previous equation (42) will cancel out in the combination of the forward and backward filter.

7.2.3 The combination step

The combination of the forward and backward filter yields the marginal smoothed distribution. Thus, for $t = \{2, \cdots, T-1\}$

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:T}) &\propto p(\mathbf{x}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{t:T}|\mathbf{x}_t) \\ &\propto \frac{p(\mathbf{x}_t|\mathbf{y}_{1:t-1})\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t:T})}{\gamma_t(\mathbf{x}_t)} \\ &\propto \frac{\int p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{x}_{t-1}|\mathbf{y}_{t-1})d\mathbf{x}_{t-1}\tilde{p}(\mathbf{x}_t|\mathbf{y}_{t:T})}{\gamma_t(\mathbf{x}_t)}, \end{aligned} \tag{44}$$

and for $t = 1$

$$p(\mathbf{x}_1|\mathbf{y}_{1:T}) \propto \frac{\mu(\mathbf{x}_1)\tilde{p}(\mathbf{x}_1|\mathbf{y}_{1:T})}{\gamma_1(\mathbf{x}_1)} \tag{45}$$

The idea is to could construct the backward filter with a finite probability measure, therefore the $\gamma_t(\mathbf{x}_t)$ in the denomenator. When the forward and backward filter are combined the effect of the artificial distribution is cancel out.

The final approximation of the combination step is a follows

$$\hat{p}(d\mathbf{x}_t|\mathbf{y}_{1:T}) \propto \sum_{i=1}^{N} \tilde{\omega}_t^i \sum_{j=1}^{N} \omega_{t-1}^j \frac{p(\tilde{\mathbf{x}}_t^i|\mathbf{x}_{t-1}^j)}{\gamma_t(\tilde{\mathbf{x}}_t^i)} \delta_{\tilde{\mathbf{x}}_t^i} d\mathbf{x}_t \tag{46}$$

7.2.4 The algorithm

To complet the derivation of Monte Carlo sampling of the two filter smoother the algorithm is given in (6)

As in the previous section an example with the TFS is given for the same setup as in the other experiments. The result of the TFS can be seen in the figure (7.2.4), note that the TFS can change the support of the posterior estimate from the forward filter and therefore the smoothed estimated is much closer to the truth. Also note the scatter plot in figure (7.1) there was some residuals from the symmetri from the likelihood kernel, which could be seen in the scatter plot as symmetri outliers. With the TFS filter these resudals have disapeared.

References

[1] Mark Briers. *Improved Monte Carlo Methods for State Space Models.* Doctoral thesis, University of Cambridge, Department of Engineering, 2007.

[2] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. Technical report, Communications Research Laboratory, McMaster University, Hamilton ON, Canada, 2003.

procedure TFS
Forward filtering
For $t = 1, \cdots, T$, perform the particle filtering to obtain the weighted measure $\{\mathbf{x}_t^i, \omega_t^i\}_{i=1}^N$

Backward filtering
1. *Initialization*
For $i = 1, \cdots, N$, Sample candidates from the proposal distribution $\mathbf{x}_t^i \sim q(\dot{|}\mathbf{y}_T)$
Compute the importance weights:
$\tilde{\omega_T}^i \propto \frac{p(\mathbf{y}_T|\tilde{\mathbf{x}}_T^i)\gamma_T(\tilde{\mathbf{x}}_T^i)}{q(\tilde{\mathbf{x}}_T^i|\mathbf{y}_T)}$
2. For $t = T, \cdots, 2$ and $i = 1, \cdots, N$
Sample candidates from the proposal distribution $\mathbf{x}_t^i \sim q(\dot{|}\mathbf{x}_t^i, \mathbf{y}_{t-1})$
Calculate the backward importance weights:
$\tilde{\omega_{t-1}}^i \propto \frac{\tilde{\omega_t}^i p(\mathbf{y}_{t-1}|\tilde{\mathbf{x}}_{t-1}^i)\gamma_{t-1}(\tilde{\mathbf{x}}_{t-1}^i p(\tilde{\mathbf{x}}_{t-1}^i|\tilde{\mathbf{x}}_t^i)}{\gamma_t(\tilde{\mathbf{x}}_t^i)q(\tilde{\mathbf{x}}_{t-1}^i|\mathbf{x}_t^i,\mathbf{y}_t)}$
Resample if necessary
end procedure

Algoritme 6: The generic two filter smoother

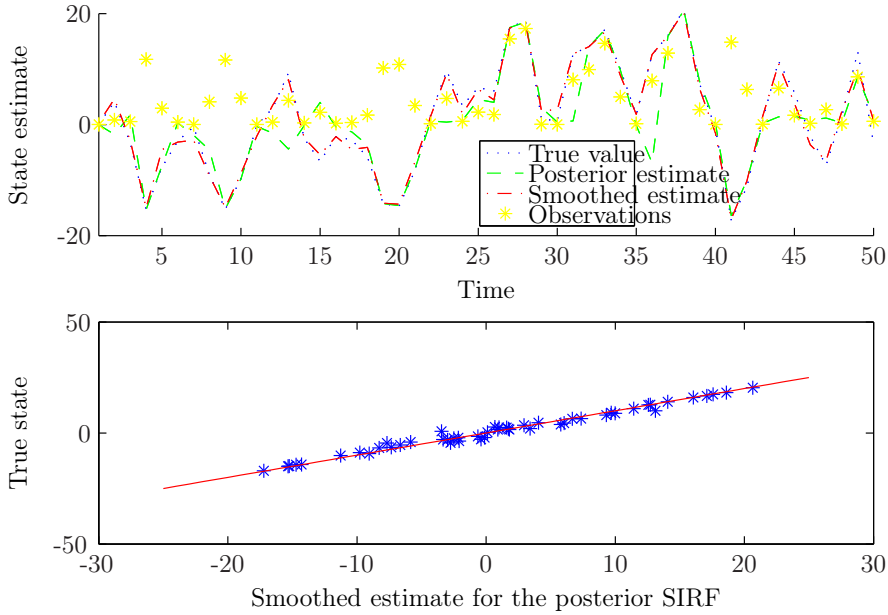


Figure 12: The smoothed estimate of the TFS and the scatter plot of the truth vs. the smoothed estimate

[3] Randal Douc, Olivier Cappe, and Eric Moulines. Comparison of resampling schemes for particle filtering, 2005.

[4] A. Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, first edition, 2001.

[5] Arnaud Doucet, Neil J. Gordon, and Vikram Krishnamurthy. Particle filters for state estimation of jump markov linear systems, 2001.

[6] John Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–1339, 1989.

[7] Jeroen D. Hol, Thomas B. Schon, and Richard Karlson. On resampling algorithms for particle filters, 2006.

[8] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[9] Mike Klass, Mark Briers, Nando de Freitas, Arnaud Doucet, Simon Maskell, and Dustin Lang. Fast particle smoothing: If i had a million particles, 2006.

[10] Augustine Kong, Jun S. Liu, and Wing Hung Wong. Sequential imputations and bayesian missing data problems. *Journal of American Statistical Association*, 89(1):278–288, 1994.

[11] Henrik Madsen. *Time Series Analysis*. DTU press, 2th edition, 2006.

[12] Henrik Madsen and Jan Holst. *Modelling Non-Linear and Non-Stationary Time Series*. DTU press, 1th edition, 2000.

[13] D. Q. Mayne. A solution of the smoothing problem for linear dynamic systems. *Automatica*, 4:73–92, 1966.

[14] A.F.M. Smith N.J. Gordon, D.J. Salmond. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140, 1993.

[15] Rudolph van der Merwe, Arnaund Doucent, Nando de Freitas, and Eric Wan. The unscented particle filter. Technical report, Cambridge University Engineering Department, 2000.

[16] Mike West. Mixture models, monte carlo, bayesian updating and dynamic models. *Computing Science and Statistics*, 24:325–333, 1993.

APPENDIX F

Paper F

Accepted for publication as IMM-Technical Report-2009-10

A parallel implementation of a finite element solver

for statistical methods

Jan Frydendall

July 21, 2009

DTU Informatics

Contents

1 Preliminary

The need for fast finite element solvers has long been a requirement for modern numerical simulation of large systems. Finite element methods is a very flexible method if the focus is on complex geometries in the domain, i.e landscape modeling, fluid dynamics, material properties modeling etc.. In the later years the finite element methods is also becoming known to statisticians who want to model more complex structures. A straight forward example is the modeling of the Fokker-Planck equation. The Fokker-Planck equation describes the time evolution of probability mass in a domain. The equation for the Fokker-Planck equation can be written as,

$$\frac{\partial p}{\partial t} + \frac{\partial (a_i p)}{\partial x_i} - \frac{1}{2}\frac{\partial^2 (b_{ij}^2 p)}{\partial x_i \partial x_j} = 0 \tag{1}$$

The Fokker-Planck equation is a diffusion process where $a(x,t)$ is the drift term and the $b^2(x,t)$ is the diffusion term with continuous sample paths. The Fokker-Planck equation completely describes the time evolution of an stochastic process X_t once it is solved [3]

The structure of the Fokker-Planck equation is similar to the well known Advection-diffusion equation of fluid dynamics [6]

$$\frac{\partial \psi}{\partial t} + \frac{\partial (u_i \psi)}{\partial x_i} - \mu \frac{\partial^2 (\psi)}{\partial x_i \partial x_j} = 0, \tag{2}$$

where ψ is the concentration of the species that is convected/advected along the path of the velocity field U and dissipated with dispersion coefficient μ.

It should then be straight forward to make a finite element implementation of the Fokker-Planck equation since much of the literature is already well covered on the Advection-Diffusion equation. However, there are some very unpleasant Gibbs phenomena that can arise with the Advection part of the solver. In order to avoid these Gibbs phenomena an artificial diffusion has to be added to the Advection part of the solver. This is known in literature as the Streamline Upwind Petrov-Galerkin (SUPG) scheme. The introduction of the SUPG scheme will be covered in Section 2 together with the basic theory of the Advection-Diffusion scheme.

The report is build up in three parts. First the basic theory is covered of the finite element methods and the Fokker-planck and the Advection-Diffusion equations together with SUPG theory. Then the sequential implementation of the Finite element solver is covered. The solver will use the $\mathbf{P_1}$ triangles, i.e. three point quadrature triangles. The implementation of the finite element solver will all be for unstructured grids for optimal flexibility. The unstructured implementation will also implicitly give solutions to structured triangle grids. The finite

element solver could with easy be extended to cover any type of geometrical structure in two dimensional grids including $\mathbf{P_2}$ triangles, $\mathbf{Q_1}$ and $\mathbf{Q_2}$ rectangles and extended to the third dimension.

The second part of the report will cover the steps of introducing sparse matrices in the FORTRAN solver. There are several books that covers the subject on implementing sparse matrices in **C**, however, it seems that one is left to ones own devices when it comes to implementing sparse matrices in FORTRAN. In Section 3 a simple and efficient way is proposed.

Finally the report covers the steps that has to be taken in order to archive efficient parallel performance from the finite element solver. The solver is implemented in FORTRAN and run on SUN Solaris Sparc architecture machines. The sun company is on of the leading developer of Symmetric Multi-Processors (SMP) computers and is on of the key players in the introduction of the OpenMP standard. The SUN Performance Library (SPL), which is a part of the SUN development studio, supports the OpenMP standard and no extra effort has to be taken to make the SPL parallel.

The code is not designed for portability and therefore the code will be parallelized in order to perform most efficiently on the SUN computers i.e. we will heavily make use of the SPL.

2 Finite element implementation

This section is not a completely introduction to the theory behind finite elements method nor the theory to Advection-Diffusion or the Fokker-Planck. The section acts only as a small introduction to the theory which should make the understanding of the solver easier.

In this report we assume that we have an incompressible fluid i.e. $\partial_x u = 0$, therefore the Advection diffusion equation simplifies to

$$\frac{\partial \psi}{\partial t} + u_i \frac{\partial(\psi)}{\partial x_i} - \mu \frac{\partial^2(\psi)}{\partial x_i \partial x_j} = 0, \tag{3}$$

2.1 Galerkin finite element method

The finite element method used in this rapport is the Galerkin finite element method. The method is very flexible once the code is developed. One can easily change to higher order method without changing the code. We only have to change the element mapping and the quadrature.

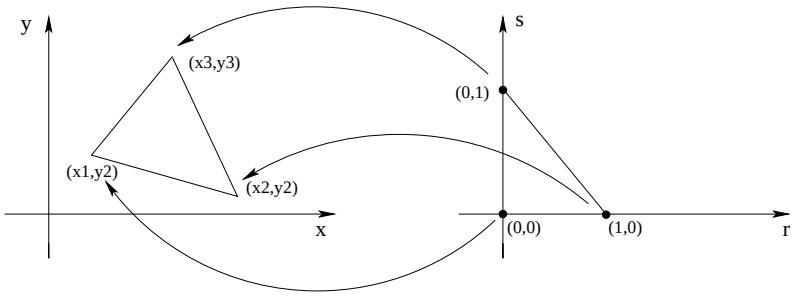


Figure 1: The isoparametric mapping of the reference $\mathbf{P_1}$ element.

Complex geometries put together from arbitrary triangles can be quit tedious to derive the derivatives from such elements. However, if we can find a mapping from a standard triangle where all derivertives are made easy to obtain the derivatives on the complex triangle. The idea is to find a mapping from a reference element in a fix coordinate system and then calculate the mapping functions. In this rapport a very brief introduction is given without the prober mathematical rigor. Consider the reference $\mathbf{P_1}$ element (the right-angled triangle) shown in Figure 2.1 together with a arbitrary element from solution space. The idea is to calculate the derivatives on the reference element and map the derivatives to the elements in the mesh.

In the given element $\mathbf{P_1}$ there are three nodes where the local derivatives can be obtain. The derivatives can found by interpolation the three nodes on the element with linear polynomials. The linear polynomials are defined as:

$$\begin{aligned} \phi_1(r,s) &= 1 - r - s \\ \phi_2(r,s) &= r \\ \phi_3(r,s) &= s \end{aligned} \tag{4}$$

With the above definition the ψ function can now be written in local coordinates:

$$\psi(r,s) = \sum_{i=1}^{3} \phi_i \psi_i \tag{5}$$

The global coordinates are found from the isoparametric mapping; first consider the coordi-

nates x_i and y_i at the nodal points the mapping in the reference element is given as

$$x(r,s) = \sum_{i=1}^{3} \phi_i x_i \tag{6}$$
$$y(r,s) = \sum_{i=1}^{3} \phi_i y_i$$

Using the chain rule the Jacobian is:

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial}{\partial s}\frac{\partial s}{\partial x} \tag{7}$$
$$\frac{\partial}{\partial y} = \frac{\partial}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial}{\partial s}\frac{\partial s}{\partial y},$$

collecting the terms in matrix form:

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \mathbf{J^{-1}} \begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial s} \end{bmatrix}, \tag{8}$$

where $\mathbf{J}$ is defined as:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{bmatrix} \tag{9}$$

Multiplying (3) with the arbitrary test function $\bar{\psi}$, the following expression is obtained, written out in fully coordinates:

$$\int_\Omega \bar{\psi}\left[u\frac{\partial \psi}{\partial x} + v\frac{\partial \psi}{\partial y} - \mu\frac{\partial^2 \psi}{\partial x \partial x} - \mu\frac{\partial^2 \psi}{\partial y \partial y}\right] dxdy = 0, \tag{10}$$

for the diffusion process the divergence and Green's theorems are used where the boundary integral is set to zero, thus,

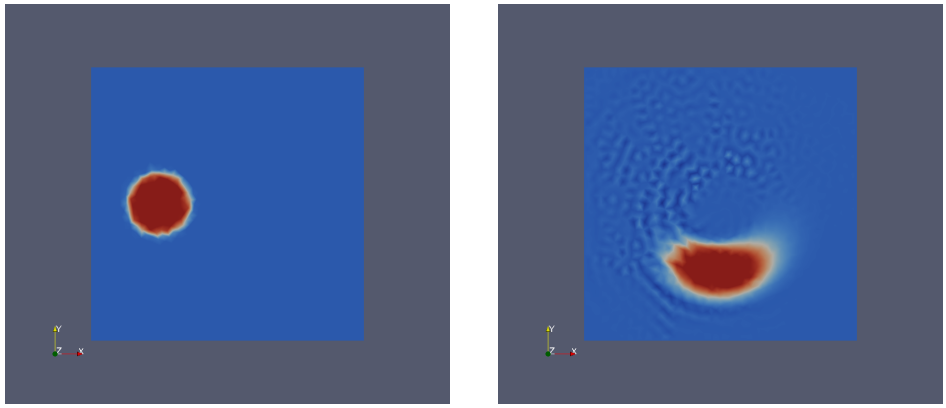
$$\int_\Omega \left[\bar{\psi}u\frac{\partial \psi}{\partial x} + \bar{\psi}v\frac{\partial \psi}{\partial y}\right] dxdy - \int_\Omega \mu\left[\frac{\partial \psi}{\partial x}\frac{\partial \bar{\psi}}{\partial x} + \frac{\partial \bar{\psi}}{\partial y}\frac{\partial \psi}{\partial y}\right] dxdy = 0, \tag{11}$$

Now it is a simple matter of finding the integration mapping by using the chain rule (8) the final equation becomes

$$\int_\Omega \left[\bar{\psi}u\frac{\partial \psi}{\partial r} + \bar{\psi}v\frac{\partial \psi}{\partial s}\right] drds - \int_\Omega \mu\left[\frac{\partial \psi}{\partial r}\frac{\partial \bar{\psi}}{\partial r}\mathbf{J}^{-1} + \frac{\partial \bar{\psi}}{\partial s}\frac{\partial \psi}{\partial s}\mathbf{J}^{-1}\right] drds = 0, \tag{12}$$

2.2 Advection-Diffusion

The Advection-Diffusion partial differential equation is a hyperbolic differential equation which are notoriously difficult to solve numerically. If a front is advected along the underlining current then Gibbs phenomena will arise around the front. Gibbs phenomena are small oscillations around the edges of sharp gradients, like that of a front. Depending on the system these oscillations will in worst case eventually deteriorate or destroy the solution. In Figures (2a - 2b) and (2c - 2d) the solution to the Advection-diffusion equation is shown. The solution is a cone that is transported along a counter-clockwise rotating velocity field. At time step $t = 15$ the Gibbs phenomena are already visible and these small oscillations multiplies as the solution time increases. At time step $t = 60$ the entire domain is covered with small oscillations if the solution is integrated even further then the small oscillations would eventually destroy the solution.

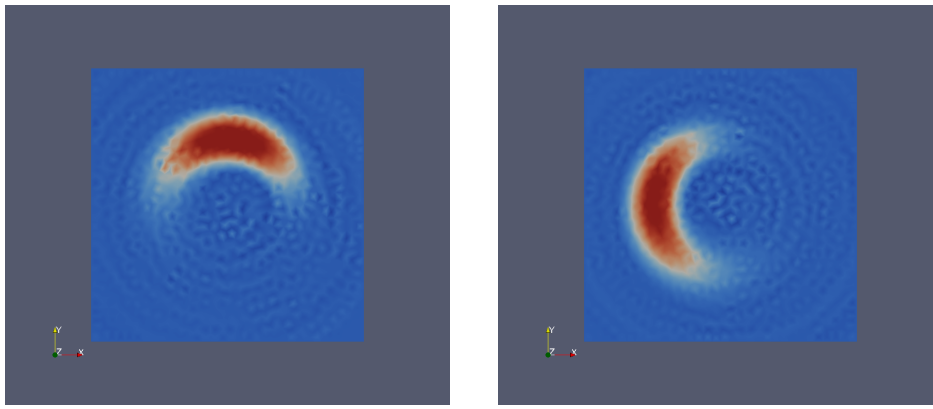


(a) t=0 **(b)** t=15

Figure 2: A cone is advected along in counter-clock vise rotating velocity field. The cone is shown at four different times $t = \{0, 15, 45, 60\}$. The solution field has many small oscillations around the cone.

To this point there are only two known solution to this problem, if the solution is an Eulerian solution. The Gibbs phenomena can be suppressed with a filter or a limiter [4]. The other option is to introduce a balancing (artificial) diffusion to blur out the Gibbs phenomena. The inspiration for the balancing diffusion is taken from the finite difference approach of the upwind scheme.

The key idea is to implement the balancing diffusion in the direction of the resulting velocity



(c) t=45 **(d)** t=60

Figure 2: A cone is advected along in counter-clock vise rotating velocity field. The cone is shown at four different times $t = \{0, 15, 45, 60\}$. The solution field has many small oscillations around the cone.

field. This will of course make the balancing diffusion anisotropic. By introducing a weighting function as [6]

$$\begin{aligned} \psi_a &= \psi_a + \alpha\psi^* \\ &\triangleq \psi_a + \frac{\alpha h}{2}\frac{U_i}{|\mathbf{U}|}\frac{\partial \psi_a}{\partial x_i}, \end{aligned} \tag{13}$$

where α is quantity that has to be calculated for each element.

$$\alpha = \coth Pe - \frac{1}{Pe}, \tag{14}$$

where

$$Pe = \frac{|\mathbf{U}|h}{2k}, \tag{15}$$

where $|\mathbf{U}| = \sqrt{U_i U_i}$. Pe is called the Peclet number and is a dimensionless quantity that relates the rate of the advection of a flow to the diffusion. The quantity h is a reasonable defineable element size. There is no real definition of how to find h, it is just some form of element measure [6]. In this case it is taken as the diameter of the element. This is done by finding the diameter of the inscribed circle of the triangle. In the book by [6] the choice of h is chosen such that the direction of h coincides with the velocity vector $\mathbf{U}$. This is not implemented for the unstructured mesh, however, for the structured mesh this is implemented in the code. The equation (13) is designed such that the balancing diffusion is only active in the direction of the flow, in the other direction the balancing diffusion should be zero.

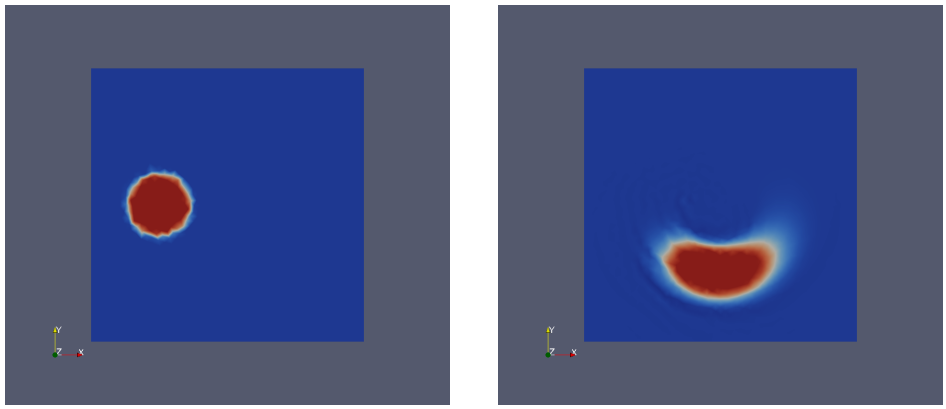
If eq. (13) is substitute into the advection part of (3) then we will get an additional diffusion term:

$$\int_\Omega \left[\bar{\psi}u\frac{\partial\psi}{\partial x} + \bar{\psi}v\frac{\partial\psi}{\partial y}\right] dxdy + \int_\Omega \left[u\bar{\psi}\frac{\alpha h}{2}\frac{u}{|\mathbf{U}|}\frac{\partial^2\psi_a}{\partial x^2} + v\bar{\psi}\frac{\alpha h}{2}\frac{v}{|\mathbf{U}|}\frac{\partial^2\psi_a}{\partial y^2}\right] dxdy \tag{16}$$

$$+ \int_\Omega \left[u\bar{\psi}\frac{\alpha h}{2}\frac{u}{|\mathbf{U}|}\frac{\partial^2\psi_a}{\partial x\partial y} + v\bar{\psi}\frac{\alpha h}{2}\frac{v}{|\mathbf{U}|}\frac{\partial^2\psi_a}{\partial y\partial x}\right] dxdy \tag{17}$$

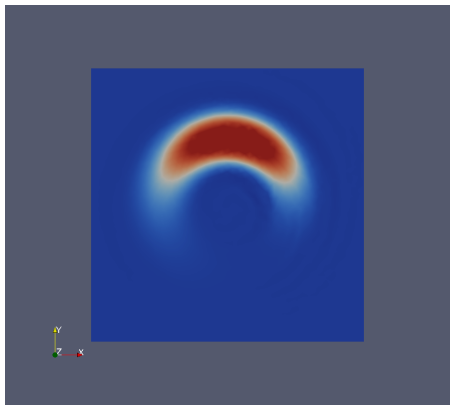
The two last terms are the balancing diffusion that suppresses the buildup of oscillations around fronts in the solver. However, the balancing can not suppress all oscillations and small oscillation may still appear in the solution. However, the oscillations are now so small that they can be neglected for the most parts. We will not go deeper into the theory of suppressing oscillations in this report. The above algorithm is implemented into the subroutine **Peclet**. We have not applied the chain rule for eq. (16) as this a trivial matter.

In Figure 3 the Streamline upwind Petrov-Galerkin method is applied to the same condition as in Figures (3a - 3b) (3c - 3d). However, the SUPG diffusion is clearly suppressing the Gibbs phenomena such that the solution is preserved. There is no indications in the figures that there any visible small oscillations.

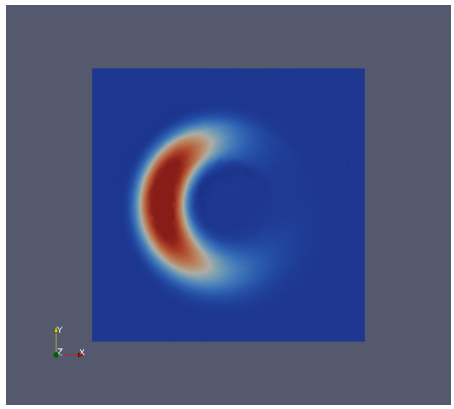


(a) t=0 **(b)** t=15

Figure 3: As in Figures (2a - 2b). With the SUPG scheme the solution is no longer prone to the Gibbs phenomena in the same extent as before.



(c) t=45



(d) t=60

Figure 3: As in Figure (2c - 2d). With the SUPG scheme the solution is no longer prone to the Gibbs phenomena in the same extent as before.

3 Sparse implementation

In order to save memory and to have a very fast code the sparsity of the finite element solver will be exploited.

3.1 The Storage formats

Choosing a sparse storage system on the other hand is more difficult. There are quite a few available. The sparskit [5] supports up to 16 different storage schemes. The different schemes are listed below.

- DNS Dense format
- BND Linpack Banded format
- CSR Compressed Sparse Row format
- CSC Compressed Sparse Column format
- COO Coordinate format
- ELL Ellpack-Itpack generalized diagonal format

- DIA Diagonal format
- BSR Block Sparse Row format
- MSR Modified Compressed Sparse Row format
- SSK Symmetric Skyline format
- NSK Nonsymmetric Skyline format
- LNK Linked list storage format
- JAD The Jagged Diagonal format
- SSS The Symmetric Sparse Skyline format
- USS The Unsymmetric Sparse Skyline format
- VBR Variable Block Row format

The storage schemes used in this report will be the COO and CSC storage format. The coordinate format or more popularly the *triplet* is by fare the easiest to work with in the assembly phase, however, it requires more memory the other more condense sparse storage schemes.

We will assemble the matrixes in the triplet format and then convert the triplet format to the CSC format before the matrices are passed to the direct solver. As a small remark, the sparse matrix structure found in matlab is also the same as the one we have chosen. Matlab uses the triplet format for communication with the user and the CSC format for all mathematical operations done behind the scenes. The triplet format is contain in two integer vectors ia and ja with respectively the i^{th} and j^{th} coordinate and one real valued vector a which stores the coordinate value k_{ij} The CSC storage format is like the triplet format also made up from two integer vectors ia and ja containing the coordinates and one real values vector a. However, the CSC format uses a more sophisticated memory efficient storage format for the coordinate vector. A small example is given to show the difference between the two storage formats. Given the matrix below the triplet format can straight forward be deducted: There are 13 non-zero elements in the matrix (18).

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 6 & 0 & 0 & 9 \\ 3 & 0 & 7 & 0 & 0 \\ 4 & 0 & 0 & 8 & 0 \\ 5 & 0 & 0 & 0 & 10 \end{pmatrix} \tag{18}$$

a	1	2	3	4	5	6	7	8	9	10
ia	1	2	3	4	5	2	3	4	2	5
ja	1	1	1	1	1	2	3	4	5	5

The CSC storage format for the matrix (18)

a	1	2	3	4	5	6	7	8	9	10
ia	1	2	3	4	5	2	3	4	2	5
ja	1	6	7	8	9	11				

The CSC storage format content is similar to the triplet format it contains the same values in the real vector a and in the row integer vector ia. However, in the column integer vector ja we now only store the pointer of each column in the vectors value and row. Thus the content of $ja(i)$ is the position in arrays a and ia where the i-th row starts [5].

The CSC storage format is well covered in literature [5, 2] and how to convert to CSC from triplet will not derived here. The sparskit toolbox [5] and the Suitesparse of [2] are respectively a FORTRAN and C library that can do these operations efficiently.

3.2 The use of sparse matrices in the code

The naive implementation of the sparse matrix will be to assemble the finite element matrices in full form and then convert them to sparse matrices once there are assembled. This approach is, however, not very efficient when we have very large data sets. With the finite element matrices in sparse form we can solve them with a standard direct sparse solver. The direct solver will covered in the next section.

However,if we want to have a fast an efficient code then we need to store the finite element matrices in sparse format from the beginning. There are a few standard libraries which can allocate the matrices in sparse format. The CS sparse toolkit from [2] is a C library that that can work with matrices in sparse form. The CS sparse toolkit is written to flawlessly interact with matlab. The code from CS sparse toolkit could be translated to FORTRAN or we could link the **C** codes directly to the FORTRAN libraries through the compiler with the respective CS sparse libraries. However, the latter can be very hazardous because of the difference in the memory structure of C and FORTRAN. To our knowledge there is no standard FORTRAN toolkit that can allocate the sparse matrices directly from within the heart of the code. It

seems that if one wants to implement a sparse structure in FORTRAN one has to start from scratch.

The first thing that is needed for sparse structure is a way of allocating the memory for the triplet structure. We need a tool to count the non-zero entries and to allocate the coordinate vectors ia and ja. When we have allocated the triplet coordinate vectors we can use them to lookup any value in the finite element matrix. This can be done with a binary search algorithm that will find the right position in the triplet format given a coordinate pair (i, j) from the given finite element matrix. The coordinate pair indicates the position of the real value of the finite element matrix entry. Once the position is located in the triplet vectors the real value is stored in the triplet value vector a at the right position. Thus, we can assemble the finite element matrices directly into the sparse structure. It is not very difficult to look up the non-zero entries if we first have the coordinate vectors and the coordinate pair. However, the difficult part arises when we want to allocate the ia and ja coordinate vectors. If we had a structured fix mesh we could once for all determined the non-zero entries by assemble the matrices in full format once and then convert to there sparse structure and count the non-zero entries and register the coordinate vectors.

However, we would like to have an adaptable unstructured finite element solver, therefore we have to find a way of counting the non-zero entries and a method to allocate the ia and ja coordinate vectors. The easiest way, but not the cleverest way, is to assemble one matrix in the preparation phase and then convert to sparse structure to get the desired information. This will work for any structure. However, this approach is very slow and will in the end take more time then the actually solving of the sparse system.

The efficient way is to count the nodal coordinates in the mesh that is used by the solver. This can be done if we have information on the adjacent nodal points in the mesh. This information is found by the subroutine **connect** in the code. The connect code is a matlab algorithm from the book [4]. The algorithm makes an element table of the mesh. The element table is a table with information of adjacent elements in the Element to Element **EToE** and the Element to Faces and **EToF**. The latter will not be used in the used in the code. For more information on the algorithm consult the book. The next step is to count the nodal points in the mesh. This is done in the code **allocate_nnz** if we have the three vertices in one triangle we can count the number of adjacent points that are connected to the respectively points. In Figure 3.2 a small structured mesh is shown. The nodal points are numbered with blue and the elements are marked with red numbers. If we look at node 25 in Table (1) we can see that the node is adjacent to the following nodal points $18, 19, 24, 25, 26, 31, 32$ including it self. So

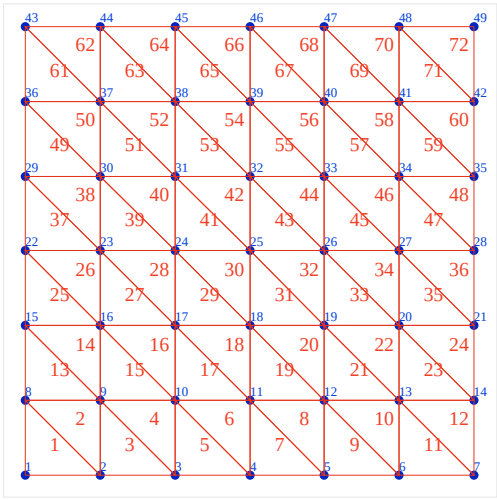


Figure 4: A small structured mesh with nodal points number with blue and the element number with red

Node	Adjacent points							Number of points
1	2	8	1					3
2	1	2	3	8	9			5
25	18	19	24	25	26	31	32	7
41	36	37	43	44				4

Table 1: A adjacent table with four nodal points which shows how the structured mesh is connected

every nodal point that is away from the boundary can at most be adjacent to seven points. In Table 1) the connectivity of the adjacent points are shown for strategic points in the mesh. Points on the boundary can have between 3 and 5 points.

To find the actual number of non-zeros is done by counting the number of adjacent points in the mesh. Start with one and then add the number of adjacent points for every node in the mesh.

The allocation of the triplet coordinate vectors is a bit trickier. Once the total number of non-zeros is known we can actual loop through the same algorithm again to store the coordinates in ia and ja. For a mesh consisting of three nodal points per triangle element we can deduct that for every element we use up to seven points in the integration of the element. Therefore it is a matter of book keeping storing the coordinates. If we look at Table (1) we can write

the coordinates for node 1 as $(1,2),(1,8),(1,1)$ and for node 2 $(2,1),(2,2),(2,3),(2,8),(2,9)$. This is done the algorithm **allocate_RowsCols**. We finally have to find a lookup table to store arbitrary coordinates (i,j) with the a_{ij}. Once we have allocated the ia and ja we can use a binary search to lookup the corresponding a_{ij} in the mesh. This is done in the algorithm $lookup_k$.

The above algorithms **allocate_nnz** and **allocate_RowsCols** are inspired by the technical report [1] and in the code found on John Burkardt homepage. The software on the homepage is free under the GNU lpgl licens. I have made my own interpretation of the code so they suit my purposes. In my search for appropriated sparse implementation in FORTRAN his homepage was the only place where I was able to find something about sparse implementation of finite element methods in Fortran. As mention before there several ways of doing this in C and C++.

4 OpenMp implementation

To facilitate the code for high performance capability of modern SMP computer we have to consider a parallel implementation of the code. We will use the OpenMP FORTRAN API for the parallelization of the code. The choice for OpenMP over MPI is because we want the code to be portable to a least Solaris computers with the SUN STUDIO compilers. The OpenMP API enables us to still maintain the code on computers without multithread support and we don't have to rewritten the entire code to get good performance. In theory OpenMP API can be implemented into the code without any modifications to the existing code. However, in practice it sometimes necessary to rewrite some parts of the code in order to get good performance.

Before we start on the implementation we have to stress out that we will not be able to get linear scaling with the entire code. There are too many synchronization in the subroutines. However, there is still a substantial amount of performance to get from the parallelization of the code and we aspect to get very good performance with the assembly of the system matrix and the direct solver.

4.1 Construction of the OpenMP clauses

The OpenMP standard backbone is the **!$omp do** clause. Since much of scientific computer codes is about crunching huge matrices in for/do loops. This clause will be used intensively

throughout the code. The parallelism archive with the standard **!$omp do** clause will not be described in this report.

The main program first calls the subroutine **read_mesh**. This subroutine reads in the mesh from the supplied mesh files, i.e. the mesh vertices and mesh elements. We can not make this code run in parallel since the file can only be accessed sequential in FORTRAN. The next subroutine called is the **allocate_sparse** this is a small routine that allocates the sparse triplet format. This subroutine calls a set of new subroutines to calculate the connectivities of the elements **connect** and the assessment of the non-zeros **allocate_nnz** and finally the allocation of the rows and columns of spares triplet format **allocate_RowsCols**. All three subroutines have been made parallel and the implementation was straight forward. In the regions where the algorithms are counting we have used the **!$omp critical** clause for the synchronizing of the threads.

However, in the subroutine **allocate_RowsCols** there is a special case where we normally could not use an omp do clause. This is because the code segment is sequential. When we do the **!$omp do** in the code shown in the Figure 5 the connectivity table is divided into n number of threads segments. Each of these segments are then calculated in the do loop. However, this would give erroneous results because we are allocating the coordinates for row and columns vectors. The counting of the coordinates is no longer in an ordered form. However, we are saved in the end of the program, because we have to sort the vectors anyway to have a faster look up table for the binary search algorithm. Therefore we can after all make an efficient parallel subroutine without an extra overhead for a sort algorithm.

After the allocation of the triplet sparse coordinate vectors the **driver** subroutine is called. In this subroutine the actual assembly of the matrices is performed.

Since the assembly is a four nested loops that runs over a set of dense sub matrices. However, we have to be careful since in a finite element assembly we actually sum the elements that are occurring more then once. Therefore we have to consider the reduction clause. In FORTRAN we have the possibility of using the reduction clause for arrays as for scalars. In **C** and **C++** we can only use the Reduction clause for scalars. The reduction clause will put the appropriated OpenMp critical clauses around the summation of the elements. A snip of the code from the subroutine **assembleConT3** can be seen in the Figure 6. The code segment also calls the binary search function **lookup_k** we have not parallelized this function since it is called from each parallel segment.

The reduction clause should be very efficient with arrays in FORTRAN. However, this im-

```
!$omp parallel if(par%npar>4) num_threads(4) private(i,k,j,t,e) default(shared)
!$omp do
! When excuting this algorithm we can not be sure that adjcopy is
! count up as in the sequential case. However, this will not
! influence the result of the subroutine. At the end of the subroutine
! we do sort the triplet, so the end result does match
! that of the sequential algorithm.
do i=1,mesh%nel
    j=EToE(i,:)==i
    if(any(j)) EToE(i,pack((/1,2,3/),mask=j))=-1
    t=EToE(i,:)
    e=mesh%EToV(i,:)
    do k=1,3
        if( i<t(k) .or. t(k)<0) then
        !$omp critical
        triplet%row(adjcopy(e(l(k,:))))  = e(l(k,:))
        triplet%col(adjcopy(e(l(k,:))))  = e(lm(k,:))
        adjcopy(e(l(k,:))) = adjcopy(e(l(k,:))) + (/1,1/)
        !$omp end critical
        end if
    end do
end do
!$omp end do
!$omp workshare
tempsort=triplet%row*expnns+triplet%col
p=(/(i,i=1,triplet%nnz)/)
p1=(/(i,i=1,mesh%nnd*mesh%nnd*mesh%nel)/)
p2=(/(i,i=1,mesh%nnd*mesh%nel)/)
!$omp end workshare nowait
!$omp single
call sortv(tempsort,1,p)
!$omp end single
!$omp workshare
triplet%col=triplet%col(p)
!$omp end workshare nowait
!$omp end parallel
```

Figure 5: A section of the **Allocate_RowsCols** code with the sort trick

```
!$omp parallel private(iel,krow,ii,kcol,jj,ival) default(shared)
!$omp & reduction(+: K,C,A,f)
!$omp do
        do iel=1,mesh%nel
            do krow=1,nnd
                ii=mesh%EToV(iel,krow)
                do kcol=1,nnd
                    jj=mesh%EToV(iel,kcol)
                    call lookup_k(triplet%row, triplet%col, ii, jj,ival)
                    K(ival) = K(ival) + ke(iel,krow,kcol)
                    C(ival) = C(ival) + ce(iel,krow,kcol)
                    A(ival) = A(ival) + ae(iel,krow,kcol)
                end do
                f(ii) = f(ii) + fe(iel,krow)
            end do
        end do
!$omp end do nowait
!$omp end parallel$
```

Figure 6: A section of the assemple code with the reduction cluse

plementation had the opposite effect in this code. In Table (2) we see the timings from the assemble code. There is a speedup from 1 to 2 threads and again from 2 to 4 threads. Increasing the number of threads from 4 to 8 the speedup starts to decrease. At 64 threads the code segment actually takes longer to be executed then the it would in the sequential case.

This is due to the fact that the reduction schedule has to synchronize all threads at the end of the loop. If the values, as in this case, are scatter over the arrays then each thread has to wait for each other to be synchronized. This generates a huge overhead. The solution to this problem is divided the domain into n-thread sub-domains. In each sub-domain the matrices can be assemble without the use of the reduction schedule.

To decompose the domain into n-thread sub-domains we will have to store the array elements that will be used in the assembly of the matrices. From the Figure 6 we can see that **ival** is the index that has to be reduced in the OpenMP section. In the subroutine **Allocate_RowsCols** we have stored all the variables $\{iel, krow, kcol, ival\}$ into an array **par%matrix** and sorted the indexes after the *ival* index. The same **ival** index will maximum occur 9 times in the

Number of threads	Execusion time
1	44.3
2	29.1
4	20.9
8	21.1
16	30.0
32	54.1
64	192.0

Table 2: Timings from the reduction assemble code

entire array. The trick is to divided the domain into n-thread sub-domains by partitioning the **ival** in the **par%matrix** such that there are no overlappings in **ival** sequences, e.g. if we have a sequences of numbers {21 21 22 22 22} then we have to make sure that the partition is made between {21|22} and not between {22|22}. With this partition we can assemble the code without the need of a reduction schedule. In Figure 7 the OpenMP loop is demonstrated.

From the Figure 7 we also see that we have made the same partition for the right hand side vector **f**. The code for the partitioning can be found in the Appendix.

On all other constructions will use the OpenMP workshare clause which is also exclusive to FORTRAN. The workshare clause supports the semantics of the FORTRAN 90 array syntax structure, since much of the code is written the FORTRAN 90 array syntax. In the subroutine **Peclet** all the array calculation is done with FORTRAN 90 array structure. A snip from the code can be seen in Figure 8.

The effect of the parallel code executed on four threads can be seen in Figure 4.1. The only part that is not running on more then one thread is the **read_mesh** subroutine. From the figure we can see that all threads are doing work, however, there are some small section in the time line which some or all threads are idle. This is due to the swicth between the different subroutines in the code. There is really nothing to be done in these parts, however, this will of course add time to the overall timing of the code.

With the tuning of the code for parallel operation we are now ready to test the code and to measure the improvements obtained from the parallelization. This will be described in the next section.

```
!$omp parallel private(th,iel,krow,ii,kcol,jj,ival,i) default(shared)
!$omp do
          do th=1,par%npar
            do i=par%startblock(th,1),par%endblock(th,1)
                iel=par%matrix(i,1);krow=par%matrix(i,2)
                kcol=par%matrix(i,3);ival=par%matrix(i,4)
                K(ival) = K(ival) + ke(iel,krow,kcol)
                C(ival) = C(ival) + ce(iel,krow,kcol)
                A(ival) = A(ival) + ae(iel,krow,kcol)
            end do
            do i=par%startblock(th,2),par%endblock(th,2)
                iel=par%vector(i,1);krow=par%vector(i,2)
                ii=par%vector(i,3)
                f(ii) = f(ii) + fe(iel,krow)
            end do
          end do
!$omp end do nowait
!$omp end parallel
```

Figure 7: A section of the new assembly matrix code. The code uses a partition scheme to implement the summation of the dense matrices into the global matrices

5 Performance of the OpenMP code

To test the potential performance that can be archive with the parallel code we will test the code with various threads numbers. The setup of the solver will be based on a large mesh and with a constant rotating velocity field. The rotating velocity field will ensure us that cone the adverting cone stays inside the domain under the test. The rotation test is also classic test for the Advection algorithm. We will not come into the details on the rotation test in the rapport since it is not the main topic. In the test we assemble the finite element matrices and solve it for 50 time steps. We will do this on a set of different threads $\{1, 2, 4, 8, 16, 32\}$ to find the limit of the speedup that can be archived with the code. All the performance will be evaluated on the SUN-HPC Euler server. We have used the SUN Grid Engine to submit the jobs to the Grid Engine.

```
! Calculate the Peclet number
! computation of element Peclet number (at the centroid)
!$omp workshare
nl12=sqrt((x(:,1)-x(:,2))**2.0_dp+(y(:,1)-y(:,2))**2.0_dp)
nl13=sqrt((x(:,1)-x(:,3))**2.0_dp+(y(:,1)-y(:,3))**2.0_dp)
nl23=sqrt((x(:,2)-x(:,3))**2.0_dp+(y(:,2)-y(:,3))**2.0_dp)
sper=(nl12+nl13+nl23)/2.0_dp
flow_l2 = sqrt(flowx(:) * flowx(:) + flowy(:) * flowy(:))
area=sqrt(sper*(sper-nl12)*(sper-nl13)*(sper-nl23))
!$omp end workshare
if(all(flowx==0.0_dp)) then
        flow_h=nl23
else if( all(flowy==0.0_dp)) then
        flow_h=nl13
else
!$omp workshare
        flow_h=area/sper
!$omp end workshare
end if
```

Figure 8: A section of the subroutine Peclet code with the workshare cluse

5.1 Density of the finite element system matrix

The mesh that we have selected for the test has 263169 nodes i.e. the size of the system matrices would be $263169^2 = 69257922561$, and the mesh consist of 524288 elements. The number of non-zeros in system is 1838081 and the density of the system matrix is $mu = \frac{1838081}{263169^2} \cdot 100 = 0.0027\%$. For a two dimensional system the large mesh is almost unrealistically large, however, to obtain some results that are measureable we had to go with the large mesh. For a smaller and more realistic mesh with 13000 elements the solver is incredibly fast and the timings are not reliable. However, the code could easily be modified to deal with three dimensions. If we have a mesh of 13000 elements in the plane and we want to discretizes the vertical into 15 layers then the system matrix is close to the size of the larger mesh.

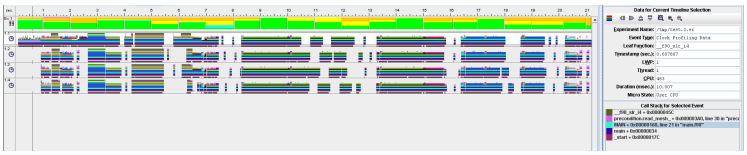


Figure 9: The computational timeline for four threads.

5.2 Parallel sparse direct solver

The solver that we have used until now in the sequential code is the direct solver provided by the SUN Performance Library (SPL). The direct solver **dgssfs** is the one-in-all interface for the direct solver. There are currently two flavors to chose from, the first is the **SP-solve** which is a FORTRAN solver with a factorization step. The other solver is **c**-based **SuperLU** direct solver. When using the **SuperLU** from a FORTRAN program we have to remember that **c**-arrays are zero-based and FORTRAN is one-based. Therefore we have to convert the triplet vectors to csc zero based. This is done with the subroutine **coo_to_csc0**.

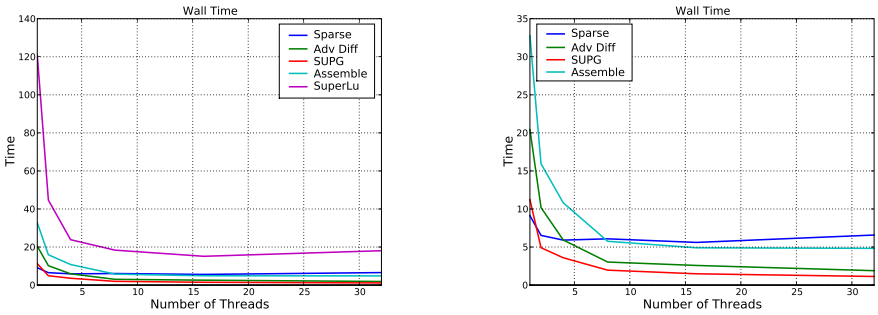
One thing that is odd with SPL is that in the documentations is stated that the **SP-solve** should be parallel. However, I have tried in many different ways to unlock this feature. However, it seems that this feature is not present in the current SUN Studio 12. It has been in previously installments of the SUN Studio series. Hopefully it will become a part of the SPL in later installments of the SUN Studio.

Since we are not able to use the build in direct solver from the SPL, we have to use another solver. We will use the **SuperLU_MT** where MT is short for Multi-thread. The **SuperLU_MT** is free software that can be downloaded from the authors homepage. The **SuperLU_MT** supports posix threads and pthreads, Solaris threads and OpenMP. However, we have not been able to get the OpenMP support to work. We therefore compile the **SuperLU_MT** to support Solaris threads. From an architectural point-of-view there is no difference between Solaris threads and OpenMP threads. It is just a matter of book keeping; and of course the easy OpenMP interface is substitute with the interface of the Solaris threads which is more complicated. However, since the **SuperLU_MT** is already put together we don't need to be worry about that fact. The **SuperLU_MT** interface is compiled and a small FORTRAN module is created to control the **SuperLU_MT**. In the newest version of the **SuperLU_MT** we are also given the choice of **colamd** [2] reordering of the matrices. This should be a more efficient than the ordinary **mmd** reordering. If we only want to use the sequential version of the code we suggest to use the **SuperLU** from the SPL, since it is

more optimizes towards the Solaris platforms.

5.3 Performance results

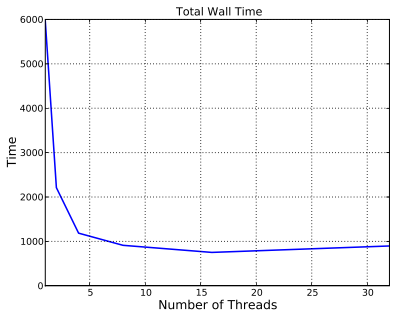
We have measured the performance at several instances in the code. Firstly we have measured the time the code is spending in the allocation part of the code. Secondly we have measured the time spend in **assembleConT3** and in the **Adv_SUPGT3** subroutines. Thirdly we have measured the overall assemble section which consists of the **assembleConT3** and in the **Adv_SUPGT3** and other small subroutine calls. Fourthly we have measured the performance of the **SuperLU** at every time step.



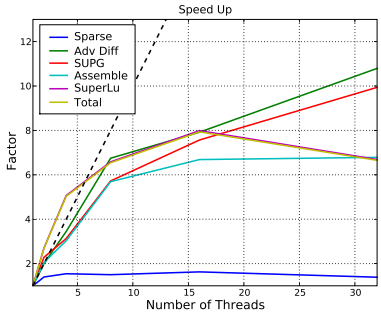
(a) wall-clock time **(b)** wall-clock time Subroutines

Figure 10: (10a) wall-clock times for all the subroutines in the code. (10b) wall-clock times for the two subroutines **assembleConT3** and the **Adv_SUPGT3**

In Figure 10a the wall-clock time is shown for the execution of the code on the different numbers of threads: $\{1, 2, 4, 8, 16, 32\}$. From the Figure 10a it is clear that the parallelism archived with OpenMP implementation is very good going from 1 thread to 2. In the beginning we see a super-scaling for all measured segments with expectance from the **Allocate** segment. We cannot hope to see any real scaling behavior from the **Allocate** segment throughout the test. This is due to the large overhead generating by multiple calls to many different small subroutines in this segment. Going from 2 to 4 threads we have linear scaling of the code segments which can be seen from Figure 10d. However, this efficiently cannot be maintained when the number of threads are increased to 8 and 16. We still see scalability, however, the efficiently is decreasing as the overhead from the OpenMP clauses are increases. There is only a minimal effect going to 32 threads. If the number of threads would be increased above 32 we will see that the measured times start to increase instead of decreasing.



(c) Total Wall-clock Time



(d) speedup

Figure 10: (10c) the wall-clock time for the overall performance of the solver. (10d) speedup archived with parallelization of the code.

In this test we have only used a constant wind field for the simulations. However, if we have had a time dependence wind field we would have had to assemble the system matrix at every time step. Therefore it is important to have a fast and scalable assembled code. In the Figures 10a 10b and 10d we see that the overall assemble segment is responding very well to the parallel implementation. Decomposing the Allocation segment into the two main components **assembleConT3** and **Adv_SUPGT3** we see that these two subroutines scales very good compared to the number threads used in the execution of the code. In section 4.1 we used much effort to get the assemble code to scale. The reduction clause where the easiest to implement put proved to scale very badly after 4 threads. First after rewriting the assemble algorithm did we get a algorithm that would scale beyond 4 threads i.e. Figure 7. However, looking at Figure 10c we see that the overall time is used by the direct solver. Therefore it is important that the solver scales good with the number of threads used in the execution of the code. The **SuperLU_MT** solver behaves very good in the range from 2 to 16 threads which can be seen from the Figure 10a, here we have only shown the mean of 49 time steps iteration. The overall time used by the code is shown in Figure 10c this figure is created by adding the different segments to a total time (there are some problems with the measuring at the HPC center at this time). This means that the **SuperLU_MT** solver is called 49 times and this segments is the most dominating part of the execution time. The choice for the **SuperLU_MT** solver was that we wanted a clean multi-thread direct solver. There are of course many other solvers on the market. If we wanted the absolutely the best performance we should consider a hybrid between OpenMP and MPI.

5.4 Conclusion

The conclusion to the OpenMP implementation of the code is that we have archived an overall execution time that is gone done by a factor 8 which can be seen from the Figure 10d if we use 16 threads. This is a very good performance and this is best that we can hope for with an OpenMP implementation. If we wanted to have a better performance on the overall execution time we should consider MPI or hybrid code between OpenMP and MPI. However, on of strengths with OpenMP is that our code still can be used without OpenMP support in the compiler. This gives a very portable structure which we cannot have with MPI. In a MPI implementation we would have had to totally reconstruct the code segments to support MPI. This is very difficult and very time consuming. However, if the MPI code would have to be used extensively over a very long time period then the investment in the construction of the MPI code would be advantageous.

To round off the OpenMP implementation in this report we have calculated the parallel efficiency of the code segments. The efficiency can be calculated from $E = \frac{T_1}{pT_p}$, where p is number of threads, E is the efficiency, T_1 is execution time for on thread and T_p is the execution time for p threads. For linear scaling the efficiency should be $E = 1$ and super scaling above $E > 1$ and scaling below linear scaling $E < 1$. The efficiency gives an indication on the payoff of the number threads invested in the code. The Figure 11 shows the efficiency

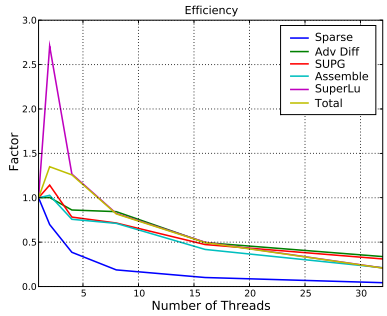


Figure 11: The efficiency for the parallel code segments as a function of the thread count.

of the parallelization. The efficiency goes below 0.5 after 16 threads after this there is no longer any payoff regarding the execution times of the code segments. Instead could the extra computational power be used for other programs. A special note on the **Allocation** segment which seems to performance very badly in all the test. In this special segment there are many calls to different subroutines. Some of these subroutines scales very well and other do not.

We have not wanted to show how each small subroutine scales in this report because is not very interesting. If someone should ever comes to use this report and the code here in we recommended to investigate the scalability of this section to convince themselves that about the scalability of the many small subroutines in this segment.

References

[1] John Burkardt. Finite element treatment of the navier stokes equations: Part vi. Technical report, School of Computational Science, Florida State University, Department of Statistics, University of Oxford, 2006.

[2] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, first edition, 2006.

[3] C.W. Gardiner. *Handbook of Stochastic Methods*. Springer Verlag, third edition, 2004.

[4] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*. Springer Verlag, first edition, 2008.

[5] Y. Saad and Research Institute for Advanced Computer Science (U.S.). *SPARSKIT: a basic toolkit for sparse matrix computations*. Research Institute for Advanced Computer Science, NASA Ames Research Center ; National Technical Information Service, 1990.

[6] O. C. Zienkiewicz, R. Taylor, and P. Nithiarasu. *The finite element method for fluid dynamics*. Elsevier, 6. edition, 2005.

A The code

A.1 Main program

```
program main
    use precision
    use precondition, only : read_mesh
    use sparse_precondition
    use solver
    implicit none
    real(dp),parameter :: theta=1.0_dp,CFL=0.9_dp,time=10.0_dp
    integer :: nns,nel,neg,nnz,steps,t,npar
    type(mesh_matrix) :: mesh
    type(COO_matrix)  :: triplet
    type(par_matrix)  :: par
    real(dp),dimension(1:2) :: viscosity
    real(dp) :: t1,t2
    character :: cpu*2
    call getarg(1,cpu)
    read(cpu,'(i2.2)')par%npar
    print *,'Number of threads: ',par%npar
    ! Call in the preconditioners from the subroutines.
    ! Allocate the arrays for the system
    t1 = omp_get_wtime()
    call read_mesh(mesh)
    !print *,'Mesh okay'
! Find the size of non-zero elements
    call allocate_sparse(par,mesh,triplet)
    !print *,'Allocation okay'
! Allocate the arrays
    viscosity=0.03_dp
    call driver(par,mesh,triplet,viscosity,CFL,theta,time)
    t2 = omp_get_wtime()
    print *,'Total time: ',t2-t1
end program main
```

A.2 Sparse allocating

```
!
! File:   sparse_precondition.f90
! Author: jf
!
```

```
! Created on January 3, 2009, 12:48 PM
!
! The subroutines are modified version from John Burkardt online libary
! https://people.scs.fsu.edu/~burkardt/f_src/f_src.html
! All credit to John Burkardt
! The subroutines have been modified to suit my purpose.
MODULE sparse_precondition
    use utils, only : print_matrixI
    use precision
    use omp_lib
    use sunperf
    use m_inssor
    contains
    subroutine allocate_sparse(par,mesh,triplet)
        implicit none
        ! Find the number of nonzero elements from the geometri of the mesh.
        !
        type(mesh_matrix),intent(inout) :: mesh
        type(COO_matrix),intent(inout) :: triplet
        type(par_matrix),intent(inout) :: par
        ! local variables
        integer, allocatable, dimension (:) :: colsum
        real(dp) :: t1,t2,t3
                !
        !  Determine the element neighbor array, just so we can estimate
        !  the nonzeros.
        !
        t3=omp_get_wtime()
        call omp_set_num_threads(par%npar)
        !t1=omp_get_wtime()
        allocate (colsum(1:mesh%nns+1) )
        call connect(mesh)
        !t2=omp_get_wtime();print *,'connect: ',t2-t1
        !print *,'Connect Okay'
        !t1=omp_get_wtime()
        call allocate_nnz(mesh,colsum,triplet)
        !t2=omp_get_wtime();print *,'Allocate_nnz: ',t2-t1
        !print *,'  Number of nonzeros : ', triplet%nnz
        !t1=omp_get_wtime()
        call allocate_RowsCols(par,mesh,colsum,triplet)
        !t2=omp_get_wtime();print *,'Allocate_RowsCols: ',t2-t1
        !print *,'Allocation of Row and Cols okay'
        !t1=omp_get_wtime()
        call boundary_node(mesh)
```

```
        !t2=omp_get_wtime();print *,'Boundary Node: ',t2-t1
        t2=omp_get_wtime()
        print *,'Allocate time:  ',t2-t3
        !
        !  Set up the sparse row and column index vectors.
        !
    end subroutine allocate_sparse

    subroutine connect(mesh)
        implicit none
        type(mesh_matrix),intent(inout) :: mesh
        !local variables
        integer,dimension(1:3*mesh%nel,1:4) :: spNodeToNode
        integer,dimension(1:3*mesh%nel,1:2) :: fnodes
        integer,dimension(1:3*mesh%nel) :: id,p,EToEv,EToFv
        integer,allocatable,dimension(:,:) :: matchL,matchR
        integer,allocatable,dimension(:) :: imatch
        integer :: Nfaces,Nnodes,i,icount,nel
        logical,dimension(1:3*mesh%nel-1) :: indices
        allocate(mesh%EToE(1:mesh%nel,1:3),mesh%EToF(1:mesh%nel,1:3))
        nel=mesh%nel
        Nfaces=3;
        Nnodes = maxval(mesh%EToV)
! create list of all faces 1, then 2, & 3
        !$omp parallel private(i) default(shared)
        !$omp workshare
        fnodes(1:nel,1:2)=mesh%EToV(:,(/1,2/))
        fnodes(nel+1:2*nel,1:2)=mesh%EToV(:,(/2,3/))
        fnodes(2*nel+1:3*nel,1:2)=mesh%EToV(:,(/3,1/))
        !$omp end workshare nowait
        !$omp do
        do i=1,3*nel
        !    call sort(fnodes(i,:))
            call inssor(fnodes(i,:))
            fnodes(i,:)=fnodes(i,:)-1
        end do
        !$omp end do nowait


! set up default element to element and Element to faces connectivity

        !$omp workshare
        mesh%EToE= spread((/(i,i=1,nel)/),nel,2)
        mesh%EToF=transpose(spread((/(i,i=1,3)/),2,nel))
```

```
! uniquely number each set of three faces by their node numbers
        id = fnodes(:,1)*Nnodes + fnodes(:,2)+1
        spNodeToNode(:,1)=id
        spNodeToNode(:,2)=(/(i,i=1,nel*3)/)
        spNodeToNode(:,3)=reshape(mesh%EToE,(/3*nel/))
        spNodeToNode(:,4)=reshape(mesh%EToF,(/3*nel/))
        p=(/(i,i=1,nel*3)/)
        !$omp end workshare nowait
        !$omp end parallel

! Now we sort by global face number.
        call sortv(spNodeToNode(:,1),1, p)
        !$omp parallel
        !$omp workshare
        spNodeToNode(:,2:4)=spNodeToNode(p,2:4)
        !$omp end workshare nowait
        !$omp end parallel
        !call print_matrixI(spNodeToNode);stop

! find matches in the sorted face list
        !$omp parallel
        !$omp workshare
        indices=spNodeToNode(1:3*nel-1,1)==spNodeToNode(2:3*nel,1)
        icount=count(indices)
        !$omp end workshare nowait
        !$omp end parallel

! make links reflexive
        allocate(matchL(1:2*icount,1:4),matchR(1:2*icount,1:4),imatch(1:icount))
        !$omp parallel
        !$omp workshare
        matchL=0;matchR=0
        imatch=pack((/(i,i=1,3*nel-1)/),mask=indices)
        matchL(1:icount,:)=spNodeToNode(imatch,:)
        matchL(icount+1:2*icount,:)=spNodeToNode(imatch+1,:)
        matchR(1:icount,:) = spNodeToNode(imatch+1,:)
        matchR(icount+1:2*icount,:)=spNodeToNode(imatch,:)

! insert matches
        EToEv=reshape(mesh%EToE,(/3*nel/));EToFv=reshape(mesh%EToF,(/3*nel/));
        EToEv(matchL(:,2)) = matchR(:,3); EToFv(matchL(:,2)) = matchR(:,4)
        mesh%EToE=reshape(EToEv,(/nel,3/));mesh%EToF=reshape(EToFv,(/nel,3/))
        !$omp end workshare nowait
```

```
    !$omp end parallel
    deallocate(matchL,matchR,imatch)

end subroutine connect

subroutine allocate_nnz(mesh,colsum,triplet)
    implicit none
    type(mesh_matrix),intent(in) :: mesh
    type(COO_matrix),intent(out) :: triplet
    integer,intent(out),dimension(:) :: colsum
    !local variables
    integer :: i,k,npar
    integer,dimension(1:mesh%nns) :: nz
    integer,dimension(1:mesh%nel,1:3) :: EToE
    integer,dimension(1:3,1:2) :: l
    integer,dimension(1:3) :: t
    logical,dimension(1:3) :: j
    real(dp) :: t1,t2
    !$omp parallel
    !$omp workshare
    EToE=mesh%EToE
    !$omp end workshare nowait
    !$omp single
    npar=omp_get_num_threads()
    !$omp end single
    !$omp end parallel
    nz=1
    l(1,:)=(/1, 2/)
    l(2,:)=(/2, 3/)
    l(3,:)=(/3, 1/)
    !$omp parallel if(npar>4) num_threads(4) private(i,k,j,t) default(shared)
    !$omp do
    do i=1,mesh%nel
        j=EToE(i,:)==i
        if(any(j)) EToE(i,pack((/1,2,3/),mask=j))=-1
        t=EToE(i,:)
        do k=1,3
            if( i<t(k) .or. t(k)<0) then
            !$omp critical
            nz(mesh%EToV(i,l(k,:)))=nz(mesh%EToV(i,l(k,:)))+1
            !$omp end critical
            end if
        end do
    end do
```

```
        !$omp end do nowait
        !$omp end parallel
        colsum=cumsumI((/1,nz/))

        triplet%nnz=colsum(mesh%nns+1)-1

    end subroutine allocate_nnz

    subroutine allocate_RowsCols(par,mesh,colsum,triplet)
        implicit none
        type(mesh_matrix),intent(in) :: mesh
        type(COO_matrix),intent(inout) :: triplet
        type(par_matrix),intent(inout) :: par
        integer,intent(in),dimension(:) :: colsum
        !local variables
        integer :: i,k,node,nnz,iel,krow,kcol,ii,jj,icount
        real(dp) :: expnns
        integer,dimension(1:mesh%nel,1:3) :: EToE
        integer,dimension(1:mesh%nns) :: adjcopy
        integer,dimension(1:3,1:2) :: l,lm
        real(dp),dimension(1:triplet%nnz) :: tempsort
        integer,dimension(1:triplet%nnz) :: p
        integer,dimension(1:mesh%nnd*mesh%nnd*mesh%nel) :: p1
        integer,dimension(1:mesh%nnd*mesh%nel) :: p2
        integer,dimension(1:3) :: t,e
        logical,dimension(1:3) :: j
        nnz=triplet%nnz
        allocate(triplet%row(1:nnz),triplet%col(1:nnz),triplet%values(1:nnz),par%vector(1:mesh%nel*mesh%
        allocate(par%matrix(1:mesh%nnd*mesh%nnd*mesh%nel,1:mesh%nnd+1),par%startblock(1:par%npar,1:2),pa
        EToE=mesh%EToE
        triplet%row=-1
        triplet%col=-1
        adjcopy=colsum(1:mesh%nns)
        expnns=10.0_dp**ceiling(log10(real(mesh%nns)))
        !$omp parallel if(par%npar>4) num_threads(4) private(node) default(shared)
        !$omp do
        do node = 1, mesh%nns
            !$omp critical
            triplet%row(adjcopy(node)) = node
            triplet%col(adjcopy(node)) = node
            adjcopy(node) = adjcopy(node) + 1
            !$omp end critical
        end do
        !$omp end do nowait
```

```
!$omp end parallel
l(1,:)=(/1, 2/);lm(1,:)=(/2, 1/)
l(2,:)=(/2, 3/);lm(2,:)=(/3, 2/)
l(3,:)=(/3, 1/);lm(3,:)=(/1, 3/)
!$omp parallel if(par%npar>4) num_threads(4) private(i,k,j,t,e) default(shared)
!$omp do
! When excuting this algorithm we can not be sure that adjcopy is
! count up as in the sequential case. However, this will not
! influence the result of the subroutine. At the end of the subroutine
! we do sort the triplet, so the end result does match
! that of the sequential algorithm.
do i=1,mesh%nel
    j=EToE(i,:)==i
    if(any(j)) EToE(i,pack((/1,2,3/),mask=j))=-1
    t=EToE(i,:)
    e=mesh%EToV(i,:)
    do k=1,3
        if( i<t(k) .or. t(k)<0) then
        !$omp critical
        triplet%row(adjcopy(e(l(k,:))))  = e(l(k,:))
        triplet%col(adjcopy(e(l(k,:))))  = e(lm(k,:))
        adjcopy(e(l(k,:))) = adjcopy(e(l(k,:))) + (/1,1/)
        !$omp end critical
        end if
    end do
end do
!$omp end do
!$omp workshare
tempsort=triplet%row*expnns+triplet%col
p=(/(i,i=1,triplet%nnz)/)
p1=(/(i,i=1,mesh%nnd*mesh%nnd*mesh%nel)/)
p2=(/(i,i=1,mesh%nnd*mesh%nel)/)
!$omp end workshare nowait
!$omp single
call sortv(tempsort,1,p)
!$omp end single
!$omp workshare
triplet%col=triplet%col(p)
!$omp end workshare nowait

!$omp do private(iel,krow,ii,kcol,jj,k,icount)
do iel=1,mesh%nel
    do krow=1,mesh%nnd
        ii=mesh%EToV(iel,krow)
```

```
            do kcol=1,mesh%nnd
                jj=mesh%ETOV(iel,kcol)
                call lookup_k(triplet%row, triplet%col, ii, jj,k)
                icount=(iel-1)*mesh%nnd*mesh%nnd+(krow-1)*mesh%nnd+kcol
                par%matrix(icount,1:4)=(/iel,krow,kcol,k/)
            end do
            par%vector((iel-1)*mesh%nnd+krow,1:3)=(/iel,krow,ii/)
        end do
    end do
    !$end do nowait
    !$omp single
    call sortv(par%matrix(:,4),1,p1)
    call sortv(par%vector(:,3),1,p2)
    !$omp end single
    !$omp workshare
    par%matrix(:,1:3)=par%matrix(p1,1:3)
    par%vector(:,1:2)=par%vector(p2,1:2)
    !$omp end workshare nowait
    !$omp end parallel
    par%startblock=0
    par%endblock=0
    do i=1,par%npar
        ! Finding the start and end positions in the decomposed mesh.
        ! First in the matrix structure
        par%startblock(i,1)=par%startblock(i,1)+size(par%matrix,1)/(par%npar)*(i-1)+1
        par%endblock(i,1)=par%endblock(i,1)+size(par%matrix,1)/(par%npar)*i
        ! Nextin the vector structure
        par%startblock(i,2)=par%startblock(i,2)+size(par%vector,1)/(par%npar)*(i-1)+1
        par%endblock(i,2)=par%endblock(i,2)+size(par%vector,1)/(par%npar)*i
        ! If the partition is inbetween two same numbers then move the end point the left to get the
        ! to be between two different to each other numbers. i.e.
        ! 20 21 22 22
        !         |
        ! new partition
        ! 20 21 22 22
        !      |
        ! this domian decomposition will avoid the reduction clause in the assemble phase.
        !
        ! First allocate the matrix
        if(par%matrix(par%endblock(i,1),4)==par%matrix(par%endblock(i,1)-1,4).and.i<par%npar) then
            icount=count(par%matrix(par%endblock(i,1)-10:par%endblock(i,1)-1,4)==par%matrix(par%endb
            par%endblock(i,1)=par%endblock(i,1)-icount
            !print *,par%matrix(par%endblock(i,1)-10:par%endblock(i,1)+2,4)
            par%startblock(i+1,1)=-icount
```

```
            end if
            ! Allocating the vector
            if(par%vector(par%endblock(i,2),3)==par%vector(par%endblock(i,2)-1,3).and.i<par%npar) then
                icount=count(par%vector(par%endblock(i,2)-10:par%endblock(i,2)-1,3)==par%vector(par%endb
                par%endblock(i,2)=par%endblock(i,2)-icount
                !print *,par%vector(par%endblock(i,2)-10:par%endblock(i,2)-1,3)
                par%startblock(i+1,2)=-icount
            end if
        end do
!        print *,par%matrix(par%startblock(:,1),4)
!        print *,par%matrix(par%endblock(:,1),4)
    end subroutine allocate_RowsCols

    subroutine boundary_node(mesh)
        implicit none
        type(mesh_matrix),intent(inout) :: mesh
        ! Local variables
        integer :: i
        logical,dimension(1:3) :: j
        allocate(mesh%BCTYPE(1:mesh%nns),mesh%BC(1:mesh%nns))
        !$omp parallel default(shared)
        !$omp workshare
        mesh%BC=.false.;mesh%BCTYPE = 1
        !$omp end workshare nowait
        !$omp do private(i,j)
        do i=1,mesh%nel
            j=mesh%EToE(i,:)==i
            if(any(j)) then
                mesh%BC(mesh%EToV(i,pack((/1,2,3/),mask=j)))=.true.
                mesh%BCTYPE(mesh%EToV(i,pack((/1,2,3/),mask=j)))=2
            end if
        end do
        !$omp end do nowait
        !$omp end parallel
    end subroutine boundary_node


    subroutine dirichletBC( mesh, triplet, A, C,f )
        implicit none
        type(mesh_matrix),intent(in) :: mesh
        type(COO_matrix),intent(in) :: triplet
        real(dp),intent(inout),dimension(:) :: A,C,f
        ! Local Variables
        integer, parameter :: DIRICHLET = 2
```

```
    integer,dimension(1:triplet%nnz) :: ia,ja
    integer :: i,node,column,nnz
    integer,dimension(1:size(mesh%BCTYPE)) :: BCTYPE
    nnz=triplet%nnz;ia=triplet%row;ja=triplet%col;BCTYPE=mesh%BCTYPE
    !$omp parallel default(__auto)
    !$omp do
    do i = 1, nnz
        node = ia(i)
        if ( BCTYPE(node) == DIRICHLET ) then
            column = ja(i)
            if ( column == node ) then
                A(i) = 1.0_dp
                C(i) = 1.0_dp
                f(node) = 0.0_dp!node_bc(node)
            else
                A(i) = 0.0_dp
                C(i) = 0.0_dp
            end if
        end if
    end do
    !$omp end do nowait
    !$omp end parallel
end subroutine dirichletBC

subroutine lookup_k(ia,ja,i,j,k)
    implicit none
    integer, intent(in) :: i,j
    integer, intent(in),target,dimension(:) :: ia,ja
    integer, intent(out)          :: k
    ! Local Variables
    integer, pointer            :: p(:)
    integer                  :: mid,offset,l
    p => ia
    k = 0
    offset = 0
    l=0
    do while (size(p) > 0)
        mid = size(p)/2+1;l=offset+mid
        if (ia(l) < i .or. (ia(l) == i .and. ja(l) < j)) then
            p => p(mid+1:)
            offset = offset + mid
        else if (ia(l) > i .or. (ia(l) == i .and. ja(l) > j)) then
            p => p(:mid-1)
        else
```

```
            k = offset + mid    ! SUCCESS!
            return
        end if
    end do
    if(size(p)==0) k=-1
end subroutine lookup_k

function cumsumI(x)
      implicit none
      integer :: i,n
      integer,intent(in),dimension(:) :: x
      integer,allocatable,dimension(:) :: cumsumI
      n=size(x,1)
      allocate(cumsumI(1:n))
      cumsumI=0
      cumsumI(1)=x(1)
      !$omp parallel private(i) shared(cumsumI,x,n)
      !$omp do ordered
      do i=2,n
          !$omp ordered
          cumsumI(i)=cumsumI(i-1)+x(i)
          !$omp end ordered
      end do
      !$omp end do
      !$omp end parallel
end function cumsumI

subroutine coo_to_csc0(ia,ja,a,csc)
  implicit none
  integer,intent(in),dimension(:) :: ia,ja
  real(dp),intent(in),dimension(:) :: a
  type(csc_matrix), intent(inout) :: csc
  !local variables
  integer :: n,k,p,m
  integer, dimension(size(ia)) :: ja1,ia1
  integer,allocatable,dimension(:) :: w,T
  n=size(ia)
  allocate(w(0:n))
  w=0;T=0
  do k=1,n
      w(ja(k))=w(ja(k))+1
  end do
  m=count(w>0)+1
  allocate(csc%colptr(1:m),csc%rowind(1:n),csc%values(1:n),T(0:m-1))
```

```
        csc%nnz=n;csc%n=m-1
        csc%colptr(1:m)=cumsumI(w(0:m-1))
        T=csc%colptr
        ja1=ja-1;ia1=ia-1
        do k=1,n
            p=T(ja1(k))+1
            T(ja1(k))=T(ja1(k))+1
            csc%rowind(p)=ia1(k)
            csc%values(p)=a(k)
        end do
    end subroutine coo_to_csc0

    subroutine coo_to_csc1(ia,ja,a,csc)
        implicit none
        integer,intent(in),dimension(:) :: ia,ja
        real(dp),intent(in),dimension(:) :: a
        type(csc_matrix), intent(inout) :: csc
        !local variables
        integer :: n,k,p,m
        integer,allocatable,dimension(:) :: w,T
        n=size(ia)
        allocate(w(0:n))
        w=0;T=0
        do k=1,n
            w(ja(k))=w(ja(k))+1
        end do
        m=count(w>0)+1
        allocate(csc%colptr(1:m),csc%rowind(1:n),csc%values(1:n),T(1:m))
        csc%nnz=n;csc%n=m-1
        csc%colptr(1:m)=cumsumI(w(0:m-1))+1
        T=csc%colptr
        do k=1,n
            p=T(ja(k))
            T(ja(k))=T(ja(k))+1
            csc%rowind(p)=ia(k)
            csc%values(p)=a(k)
        end do
    end subroutine coo_to_csc1

     subroutine csc1_vector_mult(Ap,Ai,A,x,b)
  ! multiply  A * x = b
    integer,dimension(:),intent(in)            :: Ai,Ap
    real(dp),dimension(:),intent(in)  :: A,x
    real(dp),dimension(:),intent(out) :: b
```

```
  real(dp)  :: Aij
  integer            :: n,i,p,j

  n = size(Ap)-1
! Initialize b
  b = 0.0_dp
! b = A * x
!$omp parallel default(__auto)
!$omp do
  do j = 1,n
    do p = Ap(j), Ap(j+1) - 1
      i    = Ai(p)
      Aij  = A(p)
      b(i) = b(i) + Aij * x(j)
    enddo
  enddo
!$omp end do nowait
!$omp end parallel
  end subroutine csc1_vector_mult


END MODULE sparse_precondition
```

A.3 Assembly

```
!
! File:   precondition.f90
! Author: jf
!
! Created on January 3, 2009, 12:48 PM
!
module precondition
    use utils
    use precision
    use sparse_precondition
    use omp_lib
    !use sunperf
    contains
    subroutine read_mesh(mesh)
        implicit none
        type(mesh_matrix),intent(inout) :: mesh
        !local Variables
        integer :: i,j,k
        type(mesh_matrix) :: mesh
```

```
        open(0100,file='mesh.xy.bac',status='old')
        open(0200,file='mesh.elmtab.bac',status='old')
        read(0100,*) j, mesh%nns
        read(0200,*) k, mesh%nel
        mesh%nnd=k
        allocate(mesh%vx(1:mesh%nns),mesh%vy(1:mesh%nns),mesh%EToV(1:mesh%nel,1:3))
        do i=1,mesh%nns
            read(0100,*)mesh%vx(i),mesh%vy(i)
        end do
        do i=1,mesh%nel
            read(0200,*)mesh%EToV(i,1:k)
        end do
        close(0100)
        close(0200)
    end subroutine read_mesh
!! The Galerkin Way
    subroutine dtscale(mesh,x,y,CFL,dt)
        implicit none
        type(mesh_matrix),intent(in) :: mesh
        real(dp),intent(in)  :: CFL
        real(dp),intent(out) :: dt
        real(dp),intent(in),dimension(:,:) :: x,y
        ! loacel variables
        real(dp),dimension(1:mesh%nel) :: nl12,nl13,nl23,sper,area
        !Calculate the time scale from the CFL number
        ! computation of the dtscale number as function of the inscribed cicrle
        ! diameter as characteristic for grid to chosen timestep
        !$omp parallel
        !$omp workshare
        nl12=sqrt((x(:,1)-x(:,2))**2.0_dp+(y(:,1)-y(:,2))**2.0_dp)
        nl13=sqrt((x(:,1)-x(:,3))**2.0_dp+(y(:,1)-y(:,3))**2.0_dp)
        nl23=sqrt((x(:,2)-x(:,3))**2.0_dp+(y(:,2)-y(:,3))**2.0_dp)
        sper=(nl12+nl13+nl23)/2.0_dp
        area=sqrt(sper*(sper-nl12)*(sper-nl13)*(sper-nl23))
        !$omp end workshare
        !$omp end parallel
        dt = minval(area/sper)*1.0_dp/3.0_dp*2.0_dp/3.0_dp*CFL
    end subroutine dtscale

    subroutine DrmatrixT3(r,s,x,y,jac,invjac,phi,dphidx,dphidy)
        implicit none
        !Drmatrix evaluates derivatives of triangular shape functions
        ! In the notation of NUDG, Hestehaven et. al.
        !   input
```

```
!          r        reference x coordinate
!          s        reference y coordinate
!          x        physical x vertex coordinates
!          y        physical y vertex coordinates
!   output
!          jac       The jacobian
!          invjac    The inverse of jacobian
!          phi       The shape functions
!          dphidx    The x derivatives of phi
!          dphidy    The y derivatives of phi
real(dp), intent(in), dimension(:) :: x,y
real(dp), intent(in) :: r,s
real(dp), intent(out), dimension(:) :: dphidx,dphidy,phi
real(dp), intent(out):: jac,invjac
! Local variables
integer :: nel,nnd,iv
real(dp),dimension(1:size(x,1)) :: dphidr,dphids,phi_g
real(dp) :: dxdr,dxds,dydr,dyds
nnd=size(x,1)

! Get the shape functions, e.g. the geometric interpolation functions.
call T3shape(r,s,phi_g,dphidr,dphids)

dxdr = 0.0_dp; dxds = 0.0_dp; dydr = 0.0_dp
dyds = 0.0_dp;  jac = 0.0_dp; invjac = 0.0_dp
!
do iv = 1,nnd
    dxdr = dxdr + x(iv)*dphidr(iv)
    dxds = dxds + x(iv)*dphids(iv)
    dydr = dydr + y(iv)*dphidr(iv)
    dyds = dyds + y(iv)*dphids(iv)
end do
jac = dxdr*dyds - dxds*dydr
! check The shape of the elements if the Area is less than 1e-9 then there is
! probably something wrong
if (jac < 1e-9) then
    print *, 'Bad element ...'
    if (jac <= 0.0) then
        print *, 'error - Aborted ...'
        stop
    end if
end if
invjac = 1.0_dp/jac
!
```

```
            do iv = 1,nnd
                phi(iv)    =  phi_g(iv)
                dphidx(iv) =  dphidr(iv)*dyds - dphids(iv)*dydr
                dphidy(iv) = -dphidr(iv)*dxds + dphids(iv)*dxdr
            end do
        end subroutine DrmatrixT3

        subroutine assembleConT3(par,mesh,x,y,flowxx,flowyy,viscosity ,triplet, C, K, A, f)
            implicit none
            type(par_matrix),intent(in) :: par
            type(mesh_matrix),intent(in) :: mesh
            type(COO_matrix),intent(in) :: triplet
            real(dp),intent(in),dimension(:,:) :: x,y
            real(dp),intent(in),dimension(:) ::  viscosity,flowxx,flowyy
            real(dp),intent(out),dimension(1:triplet%nnz) :: C,K,A
            real(dp),intent(inout),dimension(:) :: f

            ! Local Variables
            integer :: nnd,nng,krow,kcol,igp,iv,i,j,ii,jj,nnz,ival,iel,th
            real(dp),dimension(1:mesh%nel,1:mesh%nnd,1:mesh%nnd) :: ke,ae,ce,tke,tae,tce
            real(dp),dimension(1:mesh%nel,1:mesh%nnd) :: fe,tfe
            real(dp),dimension(1:mesh%nel) :: rhs,flowx,flowy
            real(dp),dimension(1:mesh%nnd) :: phi,dphidx,dphidy
            real(dp),dimension(1:mesh%nnd,2) :: Gp
            real(dp),dimension(1:mesh%nnd) :: Gw
            real(dp) :: jac,invjac,rp,sp,t1,t2
            nnd=size(mesh%EToV,2)
            ! Defining the element coordinate vector and shape functions
            !$omp parallel
            !$omp workshare
            K = 0.0_dp; A = 0.0_dp; C = 0.0_dp;rhs=0.0_dp
            ke = 0.0_dp; ae = 0.0_dp; ce = 0.0_dp; fe = 0.0_dp;
            tke=0.0_dp;tae=0.0_dp;tce=0.0_dp;tfe=0.0_dp
!            flowx=flow(:,1); flowy=flow(:,2)
            !$omp end workshare nowait
            !$omp end parallel
            ! setting up the Gaussian Quadrature points
            call gaussGQ(3,Gw,Gp)
            nng=size(Gp,1)
            ! inner loop over elements
            !loop over Gauss points in this case 3 times.
            do igp = 1,nng
                rp=Gp(igp,1)
                sp=Gp(igp,2)
```

```
            call windrot(rp,sp,x,y,flowx,flowy)
            rhs = cone(rp,sp,x,y)
!$omp parallel private(iel,i,j,jac,invjac,phi,dphidx,dphidy) default(shared)
!$omp do !schedule(dynamic)
            !   evaluate derivatives and interpolate the local polynomia function
            ! over the triangle shape element.
            do iel=1,mesh%nel
                call DrmatrixT3(rp,sp,x(iel,:),y(iel,:),jac,invjac,phi,dphidx,dphidy)
                do i = 1,nnd
                    do j = 1,nnd
                        tke(iel,i,j) = viscosity(1)*dphidx(i)*dphidx(j)*invjac + &
                                       viscosity(2)*dphidy(i)*dphidy(j)*invjac
                        tce(iel,i,j) = phi(i)*phi(j)*jac
                        tae(iel,i,j) = flowx(iel)*phi(i)*dphidx(j)+ &
                                       flowy(iel)*phi(i)*dphidy(j)
                    end do
                    tfe(iel,i) = rhs(iel)* phi(i)*jac
                end do
            end do
!$omp end do nowait
            !$omp workshare
            ke=ke+tke
            ae=ae+tae
            ce=ce+tce
            fe=fe+tfe
            !$omp end workshare nowait
!$omp end parallel

        end do
!$omp parallel private(th,iel,krow,ii,kcol,jj,ival,i) default(shared)
!$omp do
          do th=1,par%npar
            do i=par%startblock(th,1),par%endblock(th,1)
                iel=par%matrix(i,1);krow=par%matrix(i,2)
                kcol=par%matrix(i,3);ival=par%matrix(i,4)
                K(ival) = K(ival) + ke(iel,krow,kcol)
                C(ival) = C(ival) + ce(iel,krow,kcol)
                A(ival) = A(ival) + ae(iel,krow,kcol)
            end do
            do i=par%startblock(th,2),par%endblock(th,2)
                iel=par%vector(i,1);krow=par%vector(i,2)
                ii=par%vector(i,3)
                f(ii) = f(ii) + fe(iel,krow)
            end do
```

```
          end do
!$omp end do nowait
!$omp end parallel
    end subroutine assembleConT3

    subroutine peclet(x,y,flowx,flowy, viscosity, epe, eph, epw)
        implicit none
        real(dp),intent(in),dimension(:,:) :: x,y
        real(dp),intent(in),dimension(:) :: flowx,flowy,viscosity
        real(dp),intent(out),dimension(:) :: epe,eph,epw
        ! loacel variables
        real(dp),dimension(1:size(x,1)) :: nl12,nl13,nl23,flow_l2,sper,area,flow_h
        integer :: i
        !Calculate the Peclet number
        ! computation of element Peclet number (at the centroid)
        ! rectangle specific calculation here
        !$omp parallel
        !$omp workshare
        nl12=sqrt((x(:,1)-x(:,2))**2.0_dp+(y(:,1)-y(:,2))**2.0_dp)
        nl13=sqrt((x(:,1)-x(:,3))**2.0_dp+(y(:,1)-y(:,3))**2.0_dp)
        nl23=sqrt((x(:,2)-x(:,3))**2.0_dp+(y(:,2)-y(:,3))**2.0_dp)
        sper=(nl12+nl13+nl23)/2.0_dp
        flow_l2 = sqrt(flowx(:) * flowx(:) + flowy(:) * flowy(:))
        area=sqrt(sper*(sper-nl12)*(sper-nl13)*(sper-nl23))
        !$omp end workshare nowait
        !$omp end parallel
        if(all(flowx==0.0_dp)) then
            flow_h=nl23
        else if( all(flowy==0.0_dp)) then
            flow_h=nl13
        else
            !$omp parallel default(__AUTO)
            !!$omp do
!            do i=1,size(x,1)
!                flow_h(i) = max(nl12(i), nl23(i), nl13(i))
!            end do
            !$omp workshare
            flow_h=area/sper
            !$omp end workshare nowait
            !print *,flow_h(1:10);stop
            !!$omp end do
            !$omp end parallel
        end if
        !$omp parallel
```

```
        !$omp workshare
        eph = flow_h
        epe = flow_h*flow_l2/2.0_dp
        epw = flow_l2
        !$omp end workshare nowait
        !$omp end parallel
        if(all(viscosity==0.0_dp)) then
            !$omp parallel
            !$omp workshare
            epe=0.5_dp
            !$omp end workshare nowait
            !$omp end parallel
        else
            !$omp parallel
            !$omp workshare
            epe = epe/sum(viscosity*0.5_dp)
            !$omp end workshare nowait
            !$omp end parallel
        end if
        epe=abs(0.5_dp*(1.0_dp-1.0_dp/epe))
        !print *, 'maximum element Peclet number is',maxval(epe)
    end subroutine peclet

    subroutine Adv_SUPGT3(par,mesh,x,y, flowxx,flowyy,viscosity,triplet, A, f)
        implicit none
        type(par_matrix),intent(in) :: par
        type(mesh_matrix),intent(in) :: mesh
        type(COO_matrix),intent(in) :: triplet
        real(dp),intent(in),dimension(:,:) :: x,y
        real(dp),intent(in),dimension(:) ::  viscosity,flowxx,flowyy
        real(dp),intent(out),dimension(:) :: A
        real(dp),intent(inout),dimension(:) :: f

        ! Local Variables
        integer :: nnd,nng,krow,kcol,igp,iv,i,j,ii,jj,nnz,ival,iel,th
        real(dp),dimension(1:mesh%nel,1:3,1:3) :: ae,tae
        real(dp),dimension(1:mesh%nel,1:3) :: fe,tfe
        real(dp),dimension(1:mesh%nel) :: rhs,epe,eph,epw,lpe,flowx,flowy
        real(dp),dimension(1:3) :: phi,dphidx,dphidy
        real(dp),dimension(1:3,2) :: Gp
        real(dp),dimension(1:3) :: Gw
        real(dp) :: rp,sp,jac,invjac,onethree=1.0_dp/3.0_dp,t1,t2
        nnd=size(mesh%EToV,2)
        ! Setting the rhs vector and flow vectors
```

```
        !$omp parallel
        !$omp workshare
        ae = 0.0_dp;fe = 0.0_dp;A=0.0_dp;rhs=0.0_dp;tae=0.0_dp;tfe=0.0_dp
!        flowx=flow(:,1); flowy=flow(:,2)
        !$omp end workshare nowait
        !$omp end parallel

        ! setting up the Gaussian Quadrature points
        call gaussGQ(3,Gw,Gp)
        nng=size(Gp,1)
        ! inner loop over elements
        !loop over Gauss points in this case 3 times.
        !t1=omp_get_wtime()
        do igp = 1,nng
            rp=Gp(igp,1)
            sp=Gp(igp,2)
            ! evaluate derivatives and interpolate the local polynomia function
            ! over the triangle shape element.
            call windrot(rp,sp,x,y,flowx,flowy)
            rhs = cone(rp,sp,x,y)
!$omp parallel private(iel,i,j,jac,invjac,phi,dphidx,dphidy) default(shared)
!$omp do
            do iel=1,mesh%nel
                call DrmatrixT3(rp,sp,x(iel,:),y(iel,:),jac,invjac,phi,dphidx,dphidy)
                do i = 1,nnd
                    do j = 1,nnd
                        tae(iel,i,j) = flowx(iel)*dphidx(i)*flowx(iel)*dphidx(j)*invjac + &
                                       flowy(iel)*dphidy(i)*flowx(iel)*dphidx(j)*invjac + &
                                       flowx(iel)*dphidx(i)*flowy(iel)*dphidy(j)*invjac + &
                                       flowy(iel)*dphidy(i)*flowy(iel)*dphidy(j)*invjac
                    end do
                    tfe(iel,i) = rhs(iel)* phi(i)*jac
                end do
            end do
!$omp end do nowait
            !$omp workshare
            ae=ae+tae
            fe=fe+tfe
            !$omp end workshare
!$omp end parallel
        end do
        !t2=omp_get_wtime();print *,'SUPG Dr Matrix: ',t2-t1
        ! Scale with the Peclet number N = N_a+ h/2*U_i/U^2*dN_a/dx
        call windrot(onethree,onethree,x,y,flowx,flowy)
```

```
        call peclet(x,y,flowx,flowy, viscosity, epe, eph, epw)
        lpe=epe*(eph/epw)
        !t1=omp_get_wtime()
!$omp parallel private(th,iel,krow,ii,kcol,jj,ival,i) default(shared)! reduction(+: K,C,A)
!$omp do
          do th=1,par%npar
!             print *,par%startblock(th,2)
!            print *,par%endblock(th,2)
            do i=par%startblock(th,1),par%endblock(th,1)
                iel=par%matrix(i,1);krow=par%matrix(i,2)
                kcol=par%matrix(i,3);ival=par%matrix(i,4)
                A(ival) = A(ival) + lpe(iel)*ae(iel,krow,kcol)
            end do
            do i=par%startblock(th,2),par%endblock(th,2)
                iel=par%vector(i,1);krow=par%vector(i,2)
                ii=par%vector(i,3)
                f(ii) = f(ii) + lpe(iel)*fe(iel,krow)
            end do
          end do
!$omp end do nowait
!$omp end parallel

    !t2=omp_get_wtime();print *,'SUPG Reduction: ',t2-t1
    end subroutine Adv_SUPGT3

    subroutine T3shape(r,s,phi,dphidr,dphids)
        implicit none
        real(dp),intent(in) :: r,s
        real(dp),intent(out),dimension(:) :: phi,dphidr,dphids
        !   shape  evaluates triangular shape functions
        !   In the notation of NUDG, Hestehaven et. al.
        !   input
        !          r         x coordinate
        !          s         y coordinate
        !   output
        !          phi        shape function
        !          dphidr     x derivative of phi
        !          dphids     y derivative of phi
        !
        phi(1) =1.0_dp-r-s
        phi(2) =    r
        phi(3) =    s
        dphidr(1) = -1.0_dp
        dphidr(2) =  1.0_dp
```

```
    dphidr(3) =  0.0_dp
    dphids(1) = -1.0_dp
    dphids(2) =  0.0_dp
    dphids(3) =  1.0_dp
end subroutine T3shape

subroutine windrot(r,s,xl,yl,flowx,flowy)
    implicit none
    real(dp),intent(in),dimension(:,:) :: xl,yl
    real(dp),intent(in) :: r,s
    real(dp),intent(inout),dimension(:) :: flowx,flowy
    ! local variables
    integer :: iv,nnd
    real(dp) :: xc,yc
    real(dp),dimension(1:size(xl,1)) :: x,y
    real(dp),dimension(1:size(xl,2)) :: dphids,dphidt,phi

    ! interpolate the wind fields at the gaussian quadratic points
    ! Find the the reference coordiantes in the reference element
    ! and build the mapping from the reference element to the phyiscal element.
    !          r         reference x coordinate
    !          s         reference y coordinate
    !          xl        physical x vertex coordinates
    !          yl        physical y vertex coordinates
    nnd=size(xl,2);x=0.0_dp;y=0.0_dp
    call T3shape(r,s,phi,dphids,dphidt)

    do iv=1,nnd
        !$omp parallel
        !$omp workshare
        x = x + phi(iv)*xl(:,iv)
        y = y + phi(iv)*yl(:,iv)
        !$omp end workshare nowait
        !$omp end parallel
    end do
    ! make a rotational wind with center in main mesh
    !$omp parallel
    !$omp workshare
    xc=(maxval(x)+minval(x))*0.5_dp
    yc=(maxval(y)+minval(y))*0.5_dp
    flowx=-(y-yc)
    flowy=(x-xc)
    !$omp end workshare nowait
    !$omp end parallel
```

```
end subroutine windrot

function cone(r,s,xl,yl)
    implicit none
    real(dp),intent(in) :: r,s
    real(dp),intent(in),dimension(:,:) :: xl,yl
    real(dp),dimension(1:size(xl,1)) :: cone
    ! local variables
    integer :: iv,nns,nnd
    real(dp) :: xc,yc,h,rc,xlt,ylt
    real(dp),dimension(1:size(xl,1)) :: x,y,rr
    real(dp),dimension(1:size(xl,2)) :: dphids,dphidt,phi

    ! interpolate the cone at the gaussian quadratic points
    ! Find the the reference coordiantes in the reference element
    ! and build the mapping from reference element to the phyiscal element.
    !          r         reference x coordinate
    !          s         reference y coordinate
    !          xl        physical x vertex coordinates
    !          yl        physical y vertex coordinates
    nnd=size(xl,2);nns=size(xl,1);x=0.0_dp;y=0.0_dp
    call T3shape(r,s,phi,dphids,dphidt)

    do iv=1,nnd
       !$omp parallel
       !$omp workshare
        x = x + phi(iv)*xl(:,iv)
        y = y + phi(iv)*yl(:,iv)
       !$omp end workshare
       !$omp end parallel
    end do
    ! make a Cone with center at xc,yc
    !$omp parallel private(iv) default(shared)
    !$omp workshare
    xlt=maxval(x)+minval(x)
    ylt=maxval(y)+minval(y)
    xc=xlt*0.25_dp
    yc=ylt*0.5_dp
    h=10000.0_dp
    rc=0.125_dp*xlt
    cone=0.0_dp
    rr=sqrt((x-xc)**2.0_dp + (y-yc)**2.0_dp)
    !$omp end workshare
    !$omp do
```

```
    do iv=1,nns
        if (rr(iv) < rc) then
            cone(iv)=h*(1.0-rr(iv)/rc)
        else
            cone(iv)=0.0_dp
        end if
    end do
    !$omp end do nowait
    !$omp end parallel
end function cone

subroutine gaussGQ(quad_num,quad_w,quad_xy)
    implicit none
    integer :: quad_num
    ! Based on the subroutine quad_rule by John Burkardt
    ! https://people.scs.fsu.edu/~burkardt/f_src/f_src.html
    real (dp) :: a,b,c,d,e,f,g,h,t,u,v,w
    real (dp), dimension(quad_num) :: quad_w
    real (dp), dimension(2,quad_num) :: quad_xy

    if ( quad_num == 1 ) then

        quad_xy(1:2,1:quad_num) = reshape ( (/ &
        1.0_dp / 3.0_dp, 1.0_dp / 3.0_dp /), (/ 2, quad_num /) )

        quad_w(1:quad_num) = 1.0_dp

    else if ( quad_num == 3 ) then

        quad_xy(1:2,1:quad_num) = reshape ( (/ &
        0.5_dp, 0.0_dp, &
        0.5_dp, 0.5_dp, &
        0.0_dp, 0.5_dp /), (/ 2, quad_num /) )

        quad_w(1:quad_num) = 1.0_dp / 3.0_dp

    else if ( quad_num == 4 ) then
        a=1.0_dp/sqrt(3.0_dp)
        quad_xy(1:2,1:quad_num) = reshape ( (/ &
        -a, -a, &
         a, -a, &
         a,  a, &
        -a,  a /), (/ 2, quad_num /) )
```

```
        quad_w(1:quad_num) = (/ 1.0_dp, 1.0_dp, 1.0_dp, 1.0_dp /)

    else if ( quad_num == 6 ) then

        a = 0.816847572980459_dp
        b = 0.091576213509771_dp
        c = 0.108103018168070_dp
        d = 0.445948490915965_dp
        v = 0.109951743655322_dp
        w = 0.223381589678011_dp

        quad_xy(1:2,1:quad_num) = reshape ( (/ &
        a, b, &
        b, a, &
        b, b, &
        c, d, &
        d, c, &
        d, d /), (/ 2, quad_num /) )

        quad_w(1:quad_num) = (/ v, v, v, w, w, w /)

    else if ( quad_num == 9 ) then

        a = 0.124949503233232_dp
        b = 0.437525248383384_dp
        c = 0.797112651860071_dp
        d = 0.165409927389841_dp
        e = 0.037477420750088_dp

        u = 0.205950504760887_dp
        v = 0.063691414286223_dp

        quad_xy(1:2,1:quad_num) = reshape ( (/ &
        a, b, &
        b, a, &
        b, b, &
        c, d, &
        c, e, &
        d, c, &
        d, e, &
        e, c, &
        e, d /), (/ 2, quad_num /) )

        quad_w(1:quad_num) = (/ u, u, u, v, v, v, v, v, v /)
```

```
        else

            write ( *, '(a)' ) ' '
            write ( *, '(a)' ) 'QUAD_RULE - Fatal error!'
            write ( *, '(a,i8)' ) '  No rule is available of order QUAD_NUM = ', &
            quad_num
            stop
        end if
    end subroutine gaussGQ

    function vtkoutput(mesh,un,t) result(ier)
        use LIB_VTK_IO
        implicit none
        type(mesh_matrix),intent(in) :: mesh
        integer,intent(in) :: t
        integer :: ier
        real(dp),intent(in),dimension(:) :: un
         !Local Variables
        integer,dimension(1:mesh%nel*3) :: connect,nctype
        character :: time*4,filename*20
        write(time,'(I4.4)') t
        filename='AdvDiff'//trim(time)//'.vtk'
        connect=reshape(mesh%EToV,(/3*mesh%nel/))-1
        nctype=5
        ier=VTK_INI('BINARY',trim(filename),'Mesh','UNSTRUCTURED_GRID')
        ier=VTK_GEO(mesh%nns,mesh%vx,mesh%vy,mesh%vx*0.0_dp)
        ier=VTK_CON(mesh%nel,connect,nctype)
        ier=VTK_DAT(mesh%nns,'node')
        ier=VTK_VAR(mesh%nns,'cone',un)
!        ier=VTK_VAR('vect',nvtx,'velocity',flowx,flowy,flowy*0.0)
        ier=VTK_END()
    end function vtkoutput
end module precondition
```

A.4 Solver

```
MODULE solver
    use precondition
    use sparse_precondition
    use utils, only : print_matrix
    use omp_lib
    use superlumt
    use sunsolver
```

```
        contains
        subroutine driver(par,mesh,triplet,viscosity,CFL,theta,time)
            implicit none
            type(mesh_matrix), intent(in) :: mesh
            type(par_matrix),intent(in) :: par
            type(COO_matrix),intent(inout) :: triplet
            real(dp),intent(in) :: CFL,theta,time
            real(dp),intent(inout),dimension(:) :: viscosity
            ! local variables
            integer :: info,t,ipivot(1:mesh%nns),ier=0,msglvl=0,i,steps
            integer,dimension(1:mesh%nel) :: offset,NCtype
            real(dp),dimension(1:mesh%nel,1:3) :: x,y ! Containg the element numbers.
            real(dp),dimension(1:triplet%nnz) :: SS,RR,A,K,C,Asupg,flowx,flowy
            real(dp),dimension(1:mesh%nns) :: un,e,b,f,fsupg
            real(dp) :: dt1,dt2,handle(150),t1,t2,t3,dt
            type(CSC_matrix) :: S,R
            t3=omp_get_wtime()
            allocate(R%values(1:triplet%nnz),S%values(1:triplet%nnz))
            R%nnz=triplet%nnz;S%nnz=triplet%nnz
            flowx=0.0_dp;flowy=0.0_dp
            call omp_set_num_threads(par%npar)
            do i = 1,3
                !$omp parallel
                !$omp workshare
                x(:,i) = mesh%vx(mesh%EToV(:,i))
                y(:,i) = mesh%vy(mesh%EToV(:,i))
                !$omp end workshare nowait
                !$omp end parallel
            end do
            call dtscale(mesh,x,y,CFL,dt)
            steps=floor(time/dt)
            !print *,'Time step is: ',dt
    ! Assemble the Arrays
            t1=omp_get_wtime()
            call assembleConT3(par,mesh,x,y,flowx,flowy,viscosity,triplet,C, K, A, f)
            t2=omp_get_wtime();print *,'assembleConT3 time: ',t2-t1
            !$omp parallel
            !$omp workshare
            A=A+K
            !$omp end workshare nowait
            !$omp end parallel
            ! If needed call the SUPG diffusion matrix
            t1=omp_get_wtime()
            call Adv_SUPGT3(par,mesh,x,y, flowx,flowy,viscosity,triplet, Asupg, fsupg)
```

```
t2=omp_get_wtime();print *,'Adv_SUPGT3 time: ',t2-t1
!$omp parallel
!$omp workshare
A=A+Asupg
f=f+fsupg
!$omp end workshare nowait
!$omp end parallel
!print *,'Assemble okay'
! Step 3, Update C -> S and A -> R
call dirichletBC(mesh, triplet, A,C,f)
!print *,'Dirichlet okay'
dt1=theta*dt
dt2=(1.0_dp-theta)*dt
!$omp parallel
!$omp workshare
SS = C - dt2*A  ! S is now S, R is still A, formula (4.35) % Step 3
RR = SS + dt*A   ! R is now R, formula (4.36)               % Step 3
!$omp end workshare nowait
!$omp end parallel
! Step 4, Prepare solving
! Coverting the sparse storage triplet COO format to the sparse storage
! format that is supported by The Super LU solver, i.e. CSC sparse storage format
! (The Super LU is a C written solver, so rember to take care of the 0 based
! array format for C vs. the 1 based array storage format.)
call coo_to_csc0(triplet%row,triplet%col, RR, R)
!print *,'triplet to csc - zero based: okay'
! Converting the Mass matrix from triplet COO storage format to csc storage format.
! Here we don't need C array storage format therefore we call the conversion routine
! with iflag=1. The Mass matrix has to be multiply with the solution vector in the time
! loop. Where we use the routine csc1_vector_mult(Ap,Ai,A,x,b) from thesparse_utils module.
! We could also have used the sparsekit routine amux, however, then we had to define to new
! row and coloum vectors. Here we just use the csc type structure.
call coo_to_csc1(triplet%row,triplet%col, SS, S)
!print *,'triplet to csc - one based: okay'
un=f
!ier=vtkoutput(mesh,un,1)
!write(0100,*)un
t2=omp_get_wtime()
print *,'Assemble time: ',t2-t3
do t = 2,50
    t1 = omp_get_wtime()
    ! Step 5
    call csc1_vector_mult(S%colptr,S%rowind,S%values,un,e)
     un=e
```

```
            ! Solve Step 10
            !  One-call routine of SPSOLVE
            !$omp parallel
            !$omp single
            call solverMT(par%npar,R,un,1,mesh%nns)
            !$omp end single
            !$omp end parallel
            !call solversun(R,un)
            t2 = omp_get_wtime()
            print *,'Solver times: ',t2-t1
            !ier=vtkoutput(mesh,un,t)
            !write(0100,*)un
        end do
    close(010)
    end subroutine driver
END MODULE solver
```

A.5 SuperLu interface

```
module superlumt
    use precision
    use omp_lib
    contains
    subroutine solverMT(nprocs,csc,U,nrhs,ldb)
        implicit none
        type(CSC_MATRIX),intent(in) :: csc
        real(dp),intent(inout),dimension(:) :: U
        integer,intent(in) :: nrhs,ldb,nprocs
        ! local variables
        integer :: n,nnz,info,nprocs
        n=size(csc%colptr)-1
        nnz=csc%nnz
        call c_bridge_pdgssv(nprocs, n, nnz, nrhs,csc%values, &
                             csc%rowind,csc%colptr,U,ldb,info)
        if(info/=0) print *,'Solution error'
    end subroutine solverMT
end module superlumt
```

A.6 Precision

```
MODULE Precision
INTEGER,PARAMETER:: dp=SELECTED_REAL_KIND(15,307)
INTEGER,PARAMETER:: qp=SELECTED_REAL_KIND(33,4931)
```

```
INTEGER,PARAMETER:: i4=SELECTED_INT_KIND(4)
INTEGER,PARAMETER:: i8=SELECTED_INT_KIND(8)
INTEGER,PARAMETER:: i10=SELECTED_INT_KIND(10)

type CSC_matrix
    real(dp), dimension(:), pointer :: values => NULL()
    integer, dimension(:), pointer :: rowind => NULL()
    integer, dimension(:), pointer :: colptr => NULL()
    integer :: status,nnz,n
end type CSC_matrix
type COO_matrix
    real(dp), dimension(:), pointer  :: values => NULL()
    integer, dimension(:), pointer   :: row => NULL()
    integer, dimension(:), pointer   :: col => NULL()
    integer :: status,nnz
end type COO_matrix
type mesh_matrix
    real(dp), dimension(:), pointer ::  vx    => NULL()
    real(dp), dimension(:), pointer ::  vy    => NULL()
    integer, dimension(:,:), pointer :: EToV  => NULL()
    integer, dimension(:,:), pointer :: EToE  => NULL()
    integer, dimension(:,:), pointer :: EToF  => NULL()
    integer, dimension(:), pointer :: BCTYPE  => NULL()
    logical, dimension(:), pointer :: BC      => NULL()
    integer :: status,nns,nel,nne,nnd
end type mesh_matrix
type par_matrix
    integer, dimension(:,:), pointer :: matrix => NULL()
    integer, dimension(:,:), pointer   :: vector => NULL()
    integer, dimension(:,:), pointer   :: startblock => NULL()
    integer, dimension(:,:), pointer   :: endblock => NULL()
    integer :: npar
end type par_matrix
END MODULE Precision
```